Lexical Simplification for Non-Native English Speakers

Gustavo Henrique Paetzold

Faculty of Engineering
Department of Computer Science
University of Sheffield

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Sheffield

October 2016

Abstract

Lexical Simplification is the process of replacing complex words in texts to create simpler, more easily comprehensible alternatives. It has proven very useful as an assistive tool for users who may find complex texts challenging. Those who suffer from Aphasia and Dyslexia are among the most common beneficiaries of such technology.

In this thesis we focus on Lexical Simplification for English using non-native English speakers as the target audience. Even though they number in hundreds of millions, there are very few contributions that aim to address the needs of these users.

Current work is unable to provide solutions for this audience due to lack of user studies, datasets and resources. Furthermore, existing work in Lexical Simplification is limited regardless of the target audience, as it tends to focus on certain steps of the simplification process and disregard others, such as the automatic detection of the words that require simplification.

We introduce a series of contributions to the area of Lexical Simplification that range from user studies and resulting datasets to novel methods for all steps of the process and evaluation techniques. In order to understand the needs of non-native English speakers, we conducted three user studies with $1,000$ users in total. These studies demonstrated that the number of words deemed complex by non-native speakers of English correlates with their level of English proficiency and appears to decrease with age. They also indicated that although words deemed complex tend to be much less ambiguous and less frequently found in corpora, the complexity of words also depends on the context in which they occur. Based on these findings, we propose an ensemble approach which achieves state-of-the-art performance in identifying words that challenge non-native speakers of English.

Using the insight and data gathered, we created two new approaches to Lexical Simplification that address the needs of non-native English speakers: joint and pipelined. The joint approach employs resource-light neural language models to simplify words deemed complex in a single step. While its performance was unsatisfactory, it proved useful when paired with pipelined approaches. Our pipelined simplifier generates candidate replacements for complex words using new, context-aware word embedding models, filters them for grammaticality and meaning preservation using a novel unsupervised ranking approach, and finally ranks

them for simplicity using a novel supervised ranker that learns a model based on the needs of non-native English speakers. In order to test these and previous approaches, we designed LEXenstein, a framework for Lexical Simplification, and compiled NNSeval, a dataset that accounts for the needs of non-native English speakers. Comparisons against hundreds of previous approaches as well as the variants we proposed showed that our pipelined approach outperforms all others.

Finally, we introduce PLUMBErr, a new automatic error identification framework for Lexical Simplification. Using this framework, we assessed the type and number of errors made by our pipelined approach throughout the simplification process and found that combining our ensemble complex word identifier with our pipelined simplifier yields a system that makes up to 25% fewer mistakes compared to the previous state-of-the-art strategies during the simplification process.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The form in which content is structured and presented has a direct impact on how accessible information is. In efforts to ensure that the ever increasing amount of information produced nowadays is available to all, guidelines for the creation of more accessible content have long been promoted. The Plain English Campaign[1] in the UK is an example. The initiative provides comprehensible guidelines on how to ensure that documents, websites and other forms of content written in English will be as simple and easily comprehensible as possible. Nonetheless, as stated by former Prime Minister Tony Blair himself, "*The Plain English Campaign has played a major role in improving the way public bodies communicate with citizens. However, there is still plenty of room for improvement - not least from politicians - so the campaign's work is far from over*". Additionally, these initiatives are limited to certain countries, languages and types of content (mostly government-related).

An alternative approach to improve accessibility is to devise technologies that can be applied to already written texts to make them easier to understand. The task of automatic Text Simplification (TS) is an example of that. It consists in adapting portions of text so that they become easier to understand by a certain target audience. It was first introduced in 1996 in the work of Chandrasekar et al. (1996). They propose several interesting applications for TS, such as helping those affected by reading disabilities to understand complex texts and improving on the performance of other NLP tasks such as Machine Translation, Text Summarisation, among others.

TS can be performed in many different levels of information. Syntactic Simplification strategies, for example, attempt to make deep transformations to a sentence's syntactic structure in order to make it easier to read and comprehend. These strategies focus mostly on operations such as sentence splitting, passive to active voice transformation, anaphoric resolution and sentence compression (Paetzold and Specia, 2013; Siddharthan, 2006). Se-

[1] http://www.plainenglish.co.uk

mantic Simplification strategies, on the other hand, aim to adapt the semantic content of a sentence or paragraph through changes in its dialog structure and/or writing style (Kandula et al., 2010; Kauchak and Barzilay, 2006).

Lexical Simplification (LS) is perhaps the most self-contained of all methods of TS. It can be formally described as the task of transforming a given sentence's words in order to make it simpler, without applying direct modifications to its grammatical or syntactic structures. In simpler terms, LS consists in finding the best candidate substitution for a target complex word, given the needs of a target audience. This can be a very challenging task since different target audiences might have very distinct needs.

Throughout the years, researchers have shown that LS plays a crucial role in TS. It has been found to be an effective way of making texts more accessible to various target audiences, such as the dyslexic (Rello et al., 2013a,b,c), aphasic (Carroll et al., 1998) and those who have low levels of literacy (Aluisio and Gasperin, 2010; Watanabe et al., 2009). It is also versatile across different languages: LS strategies have been developed for English (Carroll et al., 1998; Glavaš and Štajner, 2015; Horn et al., 2014; Paetzold, 2015b), Spanish (Bott et al., 2012; Rello et al., 2013b), Swedish (Keskisärkkä, 2012), Portuguese (Aluisio and Gasperin, 2010) and others.

Research in Psycholinguistics can explain why LS is so effective in making texts more accessible. The contributions of Hirsh et al. (1992) and Nation (2001), show that English learners need to be familiar with around 95% of a text's vocabulary in order to achieve basic comprehension and with an even higher proportion of 98% for leisure. They observed, however, that those who are familiar with the vocabulary of a text can often understand the entirety of its meaning even if the grammatical constructs used are confusing to them. These findings suggest that replacing words that are unknown to the reader, which is what a reliable lexical simplifier does, can suffice, and consequently discard the need for more complex, risky and costly simplification methods, such as Syntactic and Semantic Simplification.

Although the approaches in literature vary in nature, most of them adhere to the pipeline in Figure 1.1. Simplifiers first identify a complex word in a text, generate candidate substitutions, select the ones that fit the context in which the complex word was found, rank them according to their simplicity, and finally replace the complex word with the simplest alternative. Using this "divide and conquer" approach as opposed to joint modelling all steps, modern LS approaches have managed to obtain accuracy improvements of 42% over earlier work (Horn et al., 2014).

The biggest challenges in producing simplified versions of sentences is to ensure that they do not have grammatical errors (such as improperly inflected verbs) and that they have the same meaning as the original sentence. The study of Shardlow (2014b) shows that earlier

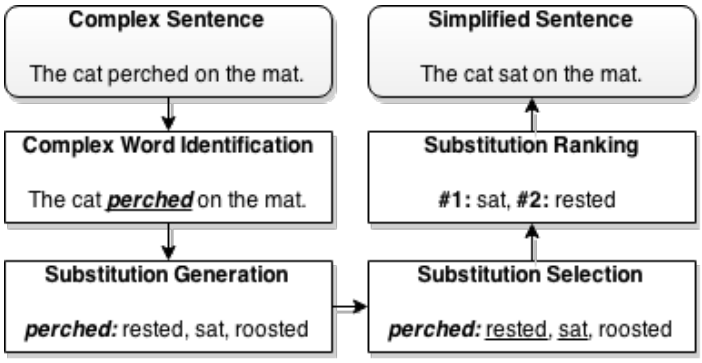

Fig. 1.1 Lexical Simplification pipeline

approaches, such as the ones of Devlin and Tait (1998) and Carroll et al. (1998), suffer from very serious limitations and tend to make several mistakes with respect to grammaticality and meaning preservation in each step of the pipeline. Even though LS approaches have greatly evolved since then, experiments with modern approaches, such as the ones reported by Biran et al. (2011), Paetzold and Specia (2013) and Glavaš and Štajner (2015), show that the quality of the simplifications produced still needs to be improved in order for LS to be employed in practice. Their results show that, despite employing much more sophisticated techniques, modern LS strategies still make some of the same simplification mistakes made by early approaches.

In this thesis, we focus on understanding and addressing the LS needs of non-native English speakers, who are numbered in hundreds of millions in the U.S alone (Lewis et al., 2016). Surprisingly, the reading challenges they face have not been thoroughly studied from an LS perspective. The scarcity of this type of work has led to a shortage of knowledge and resources about the needs of non-native speakers of English with respect to the types of reading comprehension challenges they face, hampering progress in the field.

There are also numerous limitations in modern approaches to LS. Perhaps the most outstanding of them is the weak connection between Complex Word Identification and the remaining steps in the pipeline. To our knowledge, very few simplifiers for English perform Complex Word Identification explicitly. As demonstrated by Shardlow (2014a), ignoring this step can lead to numerous unnecessary replacements that can compromise the sentence's grammaticality and/or meaning.

In order to generate candidate substitutions for complex words, most contributions still rely on resources such as complex-to-simple parallel corpora (Horn et al., 2014; Paetzold and Specia, 2013) and manually created thesauri (Baeza-Yates et al., 2015; Biran et al., 2011). Although these resources can be very useful, they can only be found in abundance for the English language, which makes these Substitution Generation approaches language-

constrained. As discussed by Shardlow (2014a), these resources also suffer from low coverage, meaning that one may not find candidate substitutions for many words which should be simplified.

Another characteristic that most LS strategies share is the absence of a dedicated Substitution Selection step. Because of the inherent complexity of Substitution Selection, most modern simplifiers ignore this task entirely, and instead employ a context-aware Substitution Ranking approach (Baeza-Yates et al., 2015; Glavaš and Štajner, 2015; Horn et al., 2014; Paetzold and Specia, 2013). Although the results reported in these contributions show that these simplifiers offer noticeable performance improvements over earlier work, they also reveal that these strategies still struggle to ensure grammaticality and meaning preservation, mostly due to the fact that their Substitution Ranking approaches alone are not able to effectively capture these linguistic properties along with simplicity.

One of the main reasons for that is the fact that most Substitution Ranking approaches for LS are unsupervised, and rely mainly on either hand-crafted metrics, or word frequency features (Baeza-Yates et al., 2015; Biran et al., 2011; Bott et al., 2012; Glavaš and Štajner, 2015; Nunes et al., 2013). Although supervised rankers require training data, they can learn how to combine different features in order to ensure that the rankings produced will fit the needs of a target audience, provided that the training data represents said needs. They can also combine features that represent different linguistic properties, such as simplicity, grammaticality and meaning preservation, which can help in the creation of more reliable joint models for Substitution Selection and Ranking. The only supervised rankers available are the ones of Jauhar and Specia (2012) and Horn et al. (2014), which use Support Vector Machines (SVM). Although sophisticated, SVM ranking was devised as a general ranking algorithm and thus has limited potential for the purposes of LS.

Another limitation in previous work refers to how the performance of simplifiers is assessed. In order to evaluate their simplifiers, most choose to either compare the performance of their strategies to that of others on an evaluation dataset, or to conduct a human evaluation of the simplifications produced. The later is the most reliable of the two, but because of its inherently high cost of both time and resources, it prevents performance comparisons from including more than a handful of simplifiers. The former, on the other hand, allows for much larger benchmarks, but is often performed over datasets created by annotators without a known background. The most frequently used evaluation datasets are LSeval (De Belder and Moens, 2012a) and LexMTurk (Horn et al., 2014), which include barely any information on the profile of the annotators who created them. Since there is no way of knowing whose needs these datasets represent, a simplifier that obtains high performance scores on these datasets is not guaranteed to help a given target audience in practice. Another problem with

the evaluation methods used in literature is that they do not incorporate Complex Word Identification, nor do they provide any insight on what the limitations of the simplifiers evaluated are, which consequently makes it difficult for one to know how they can be improved.

In the Section that follows, we discuss how we aim to address each of these limitations in LS.

1.1 Aims and Scope

1. **To provide a better understanding of the simplification needs of non-native speakers of English**: It is very important to thoroughly investigate the simplification needs of the audience being targeted before creating new technologies to assist them. This is demonstrated by Rello et al. (2013a,b,c), who perform numerous user studies with dyslexic people in order to better outline the challenges they face. Using this knowledge, they were able to create cost-effective TS approaches for them.

   We focus on studying the simplification needs of non-native speakers of the English language, an ever growing audience composed by all English speakers who have been taught a different native language. We conducted several user studies with non-native speakers of English, each focusing on a different part of the Lexical Simplification process. Through these, we produced new resources and knowledge that allowed us to create even more reliable simplifiers for non-native English speakers.

2. **To model word complexity as perceived by non-native English speakers**:

   The experiments of Shardlow (2013a) show that Machine Learning techniques can outperform the threshold-based approaches commonly used in Complex Word Identification. Although valuable, their experiments assess the performance of Support Vector Machines only.

   In order to further understand their potential, we experiment with various Machine Learning techniques, ranging from the simple Perceptron to more elaborate Neural Networks. We propose Performance-Oriented Soft Voting, an ensemble technique capable of combining systems with heterogeneous confidence estimates. Using this technique, we create a complex word identifier that outperforms all other 41 systems submitted to the Complex Word Identification shared task of SemEval 2016, in which 21 teams participated.

3. **To create effective language-agnostic Substitution Generation strategies**:

We tackle the task of Substitution Generation by exploring modern distributional semantics models, otherwise known as word embedding or word vector models. Unlike the widely used thesauri and complex-to-simple parallel corpora, these models can be trained on raw text alone, which allow them to be applied to almost any language.

We introduce a novel version of these models that exploits not only the grammatical information of the word being simplified, but also the data available in hand-crafted thesauri, such as the synonymy, hypernymy and hyponymy relations available in WordNet (Fellbaum, 1998). This way, we create a Substitution Generation strategy that can be easily adapted to various languages and also improved through the incorporation of the manually created resources available for them.

4. **To devise a novel unsupervised approach to Substitution Selection**: The few LS strategies which do employ a dedicated Substitution Selection step use some form of Word Sense Disambiguation. However, these strategies have been shown not to be very suitable for LS, since Word Sense Disambiguation is an open problem on itself (Basile et al., 2014), and requires language-constrained thesauri that often suffer from low coverage (Nunes et al., 2013; Thomas and Anderson, 2012).

   In order to address this problem, we introduce a new unsupervised approach to the task. Our strategy, which we name Unsupervised Boundary Ranking, learns a supervised model from data collected in an unsupervised fashion. It collects said data by exploring the assumption that the complex word being simplified is irreplaceable, and then uses this data to train a state-of-the-art ranker that determines how likely each candidate substitution is of fitting the context in which the complex word was found. Much like our Substitution Generation approach, this Substitution Selection strategy can be easily adapted to any language and incorporated in any LS architecture.

5. **To conceive a novel supervised Substitution Ranking approach tailored to Lexical Simplification**: The SVM-based rankers of Jauhar and Specia (2012) and Horn et al. (2014) are the only strategies in literature that employ supervised learning from data. The experiments of Glavaš and Štajner (2015) show, however, that despite being the most sophisticated approaches in literature, their SVMs can be outperformed by much simpler unsupervised approaches.

   Since supervised rankers are the only approaches that can automatically learn a model based on data that represents the needs of non-native English speakers, we focus in creating a new approach of this kind. We name our strategy Supervised Boundary Ranking. It uses the same learning framework of our Unsupervised Boundary Ranking approach to Substitution Selection, but instead of learning a model from data gathered

in an unsupervised fashion, it learns a model from human annotations made by non-native English speakers. Unlike existing SVM-based rankers, our approach can learn a state-of-the-art model from data using various Machine Learning techniques.

6. **To design more informative and cost-effective evaluation strategies for Lexical Simplification**: Although there have been numerous efforts in creating Lexical Simplification systems, not much attention has been paid when it comes to ensuring that they are properly evaluated.

   We address this problem in two ways. By re-using existing datasets and incorporating information from our user studies, we produce a new gold-standard dataset for evaluation that better captures the needs of non-native English speakers. We also introduce a new framework for the automatic categorisation of errors made by simplifiers that, unlike other evaluation methods, incorporates all steps of the typical LS pipeline.

1.2 Main Contributions

The main contributions from this thesis are:

1. Valuable new insight on the Lexical Simplification needs of non-native speakers of English. Through an array of user studies, we find that the words which challenge them the most tend to be less ambiguous and more frequently found in corpora than those which do not. We also find that a word's context offers clues with respect to its simplicity, and that word length and number of syllables have little to do with word complexity for non-native English speakers.

2. An experiment that highlights the effectiveness of pipelined LS approaches. We introduce a new LS strategy that jointly models Substitution Generation, Selection and Ranking with Neural Language Models, and compare its performance to various pipelined approaches. We find that our joint model can only perform competitively to previous work when incorporated into the typical LS pipeline.

3. Novel, state-of-the-art approaches for each and every step of the typical LS pipeline. We generate candidate substitutions using new retrofitted context-aware word embedding models, select them employing an innovative unsupervised approach that learns a model from unannotated data, and rank them using a novel supervised approach that models the needs of non-native English speakers while addressing the limitations in previous work.

4. The LEXenstein framework, which provides the implementation of dozens of LS strategies. LEXenstein is a modular and extensible Python framework with implementations of all Complex Word Identification, Substitution Generation, Selection and Ranking approaches mentioned in this thesis, and also of numerous other useful tools for LS.

5. A range of new datasets and resources to be used in the creation of new LS approaches. Through our user studies with non-native English speakers, we produced datasets that contain 211,564 annotations made by 1,000 annotators for different steps of the LS pipeline. To ensure that our novel strategies for LS achieve state-of-the-art performance in simplifying words for non-native English speakers, we also created new datasets for LS, a new corpus of subtitles of movies and series for family and children, as well as a database of automatically inferred psycholinguistic features.

6. An extensive survey and benchmark of work in LS. We describe and discuss the main contributions in literature to each step of the typical LS pipeline individually. In order to provide a better outline of the state-of-the-art in LS, we benchmark all approaches by comparing the performance of 1,344 distinct combinations of Substitution Generation, Selection and Ranking strategies.

7. A new evaluation strategy that better outlines the limitations of LS approaches. We introduce PLUMBErr: a novel method for automatic error categorisation for LS. It disregards the needs for human annotation by exploiting some of the new resources introduced in this thesis to identify numerous types of errors made by pipelined simplifiers throughout the simplification process.

8. The Complex Word Identification task of SemEval 2016, which has expanded the reach of LS in the research community. Using the dataset created in our user study on Complex Word Identification, 21 teams contributed with 42 novel approaches to identifying words that challenge non-native English speakers.

1.3 Published Material

Much of our work has been published in international conferences and workshops:

- Our thesis proposal and a summarised version of our survey was published in the Student Research Workshop at NAACL 2015 (Paetzold, 2015b).

- Our Complex Word Identification user study and shared task are the subject of a task description paper in SemEval 2016 (Paetzold and Specia, 2016f).

- The SV000gg systems, which employ our ensemble approaches to Complex Word Identification, are the subject of a system description paper in SemEval 2016 (Paetzold and Specia, 2016h).
- The SubIMDB corpus, as well as our Substitution Generation and Substitution Selection approaches were published in AAAI 2016 (Paetzold and Specia, 2016i).
- The LEXenstein framework was published as a system demonstration in ACL 2015, which is also our first contribution featuring Substitution Ranking through Boundary Ranking (Paetzold and Specia, 2015).
- Our bootstrapping technique for the inference of psycholinguistic properties of words was published as a short paper in NAACL 2016 (Paetzold and Specia, 2016c).
- A benchmarking of Lexical Simplification systems that features our Substitution Generation, Selection and Ranking strategies was published as an abstract in LREC 2016 (Paetzold and Specia, 2016b).
- The PLUMBErr framework was published as a long paper in QATS 2016 (Paetzold and Specia, 2016e).

We have also published numerous contributions that do not pertain directly to this thesis' subject:

- An algorithm for Tree Kernel calculation that employs novel Positional Suffix Trees was published as a short paper at NODALIDA 2015 (Paetzold, 2015c).
- A flexible lexical analyzer for compilers that allow for error identification was published as in the Brazilian Electronic Magazine for Scientific Innovation and Technology (Paetzold and Schemberger, 2015).
- Okapi+QuEst: a translation quality estimation extension to Okapi was published in EAMT 2015 (Paetzold et al., 2015).
- QuEst++: a multi-level quality estimation framework was published as a system demonstration in both ACL 2015 (Specia et al., 2015) and EAMT 2016 (Paetzold and Specia, 2016d).
- SHEF-NN: a set of quality estimation approaches that exploit Neural Networks was published as a long paper in WMT 2015 (Shah et al., 2015).

- The SimpleNets systems, which present supervised approaches with Recursive Neural Networks to Text Simplification and Machine Translation quality assessment was the subject of system description papers in both QATS 2016 (Paetzold and Specia, 2016g) and WMT 2016 (Paetzold and Specia, 2016a).

- SHEF-MIME: an imitation learning approach to Machine Translation Quality Assessment was published as a system description paper in WMT 2016 (Beck et al., 2016).

1.4 Thesis Structure

Chapter 2 provides a detailed outline of the state-of-the-art in Lexical Simplification, as well as a discussion of what the main limitations in each contribution are. Our survey also includes an extensive discussion on the user studies conducted with various distinct target audiences from a Lexical Simplification standpoint.

Chapter 3 describes three user studies conducted with non-native English speakers. We present the methodology used in each one of them, the findings from our data analyses and the resources produced. In this Chapter are also included the findings of the Complex Word Identification shared task of SemEval 2016, in which researchers from very diverse backgrounds have conceived strategies for identifying complex words in English sentences.

Chapter 4 introduces our joint Lexical Simplification strategy, which employs language models based on Recurrent Neural Networks to model steps of the traditional LS pipeline jointly. We show that this joint approach is not as reliable as pipelined strategies.

Chapter 5 is the first of three chapters that present our individual approaches for each step of the traditional LS pipeline. It introduces our approach for Substitution Generation, which explores the use of retrofitted context-aware word embedding models. Our models innovate by incorporating lexicon retrofitting (Faruqui et al., 2015) in a context-aware setup that allows for words to be represented by more than one vector. Our approach outperforms other generators in literature by a considerable margin.

Chapter 6 presents our approach to Substitution Ranking. It uses a flexible supervised ranking strategy that we name Boundary Ranking, in which a ranking model is learned from a binary classification setup inferred from ranking examples.

Chapter 7 describes our approach to Substitution Selection, which is the step between Substitution Generation and Ranking. We arrange our Chapters this way because our approach to Substitution Selection builds on our approach to Substitution Ranking. We take Substitution Selection to be a ranking problem, and address it using an unsupervised

Boundary Ranking approach, in which a supervised model is learned from training data acquired in an unsupervised fashion. We find that our strategy outperforms all others in literature.

Chapter 8 presents various resources, tools and benchmarks. We introduce SubIMDB: a corpus of subtitles of movies and series for family and children that aims to represent the vocabulary with which non-native English speakers are familiar. By exploiting SubIMDB as well as other resources, we propose a strategy for the automatic inference of psycholinguistic properties of words. We use these features to improve on the performance of our Substitution Ranking strategy. We then describe LEXenstein: a framework for Lexical Simplification that provides a wide array of utilities, as well as several Complex Word Identification, Substitution Generation, Selection and Ranking approaches from literature, including ours. Using LEXenstein, we present a benchmarking that compares the performance of $1,344$ simplifiers over NNSeval, a new evaluation dataset created from our user studies that better represents the needs of non-native English speakers. We then assess the performance of the best performing simplifiers using PLUMBErr: an automatic error identification framework for Lexical Simplification. The results suggest that combining our approaches to Complex Word Identification, Substitution Generation, Selection and Ranking yields a simplifier that makes the least amount of mistakes when simplifying words for non-native English speakers.

Finally, in Chapter 9 we provide our final remarks and directions for future work.

Chapter 2

Lexical Simplification: A Survey

In the last two decades, multiple approaches have been proposed to perform Lexical Simplification. Perhaps the most straightforward of them all is simplification by synonym substitution, where the main goal is to replace complex words in a given sentence by simpler synonyms. This approach was first devised by Devlin and Tait (1998), who extract synonyms from WordNet (Fellbaum, 1998) and rank candidates according to their Kucera-Francis coefficient value (Rudell, 1993).

Other approaches to Lexical Simplification combine both lexical and syntactic information to improve simplification performance. Paraphrasing strategies such as the ones by Kauchak and Barzilay (2006) and Paetzold and Specia (2013) aim to replace complex phrasal constructions with simpler alternatives. Some approaches focus on a specific knowledge domain, or on a specific class of word. Rello et al. (2013c), for example, aim to simplify numerical expressions, while Kandula et al. (2010) and Elhadad and Sutaria (2007) focus on simplifying complex medical expressions.

Even though approaches differ from one another in various ways, most of them use a very similar sequence of steps to simplify sentences. To allow for a better comprehension of the general procedures involved in LS, we define the task as the pipeline of steps illustrated in Figure 1.1, as introduced by Shardlow (2014b). The steps are the following:

1. **Complex Word Identification**: Task of deciding which words of a given sentence may not be understood by a given target audience and hence must be simplified.

2. **Substitution Generation**: Task of finding words or expressions that could replace the target complex word.

3. **Substitution Selection**: Task of deciding which of the generated candidate substitutions can replace the complex word without compromising the sentence's grammaticality nor meaning in a given context.

4. **Substitution Ranking**: Task of ranking the remaining candidate substitutions of a given complex word by their simplicity.

Previous work includes various approaches to each one of these steps throughout the years. Supervised Substitution Ranking strategies such as the one of Joachims (2002), for example, can be used not only in LS, but also any other ranking task, such as sentence re-ranking in Machine Translation and page ranking in Information Retrieval. Yet popular Text Simplification surveys, such as the ones of Siddharthan (2014) and Shardlow (2014b), tend to ignore the intricacies of LS approaches.

In an effort to address this gap in literature, the following Sections describe a survey on the many contributions presented for the various steps in the typical LS pipeline.

2.1 User Studies

One of the main goals of LS approaches in literature is to allow for people that suffer from linguistic-related illnesses and impairments to have access to a given type of content which is inherently complex to read. Consequently, a very important step in the process of developing a simplifier is assessing the needs of the target audience for which it is designed.

The two most frequently addressed target audiences in LS are Aphasia and Dyslexia. A lot of contributions in Psycholinguistics and Medicine focus on understanding those conditions, and provide valuable information about them. Because they are out of the scope of this thesis, we do not attempt to survey all the contributions that assess the difficulties and needs of those target audiences from a strictly medical standpoint. Rather, we focus only on contributions that provide discussions on how simplification can help users.

Each of the following Sections addresses a single target audience, providing a survey of the user studies conducted with the intent of understanding their needs from a simplification standpoint.

2.1.1 Aphasia

Aphasia is an acquired disorder which is usually caused by blunt traumas in certain areas of the brain, brain hemorrhages, tumors or even progressive neurological diseases, such as Alzheimer's syndrome. According to the American Speech-Language-Hearing Association[1], there have been also documented cases of Aphasia being caused by strokes and severe cases of epilepsy.

[1] http://www.asha.org/public/speech/disorders/Aphasia

According to Goodglass et al. (2001) and Devinsky and D'Esposito (2003), there are four main types of Aphasia, each associated to a distinct array of complications and symptoms. They are:

- **Non-Fluent Aphasia**: Also commonly referred to as Broca's Aphasia, it mostly affects the patient's ability to express himself, through both speaking and writing. The patient's ability to comprehend spoken and written content is usually not affected. Some of the most frequently observed impairments are the inability to pronounce long sentences in their entirety and the difficulty in constructing sentences to express elaborate thoughts.

- **Fluent Aphasia**: Frequently referred to as Wernicke's Aphasia, it can affect the patient's capability of speaking and understanding the meaning of speech, and it can also severely impair their ability of reading and writing. One of the most frequently observed symptom among patients is the inclusion of inadequate or out of context words in spoken sentences, rendering them incoherent, and sometimes leading the patient to lose their train of thought mid-sentence.

- **Anomic Aphasia**: Also known as Dysnomia, Anomic Aphasia is different than Fluent and Non-Fluent Aphasia in the sense that it rarely affects the patient's reading, writing and speaking skills directly, but rather makes it difficult for them to select the appropriate words to express a given concept. Patients are often able to pronounce words with not much difficulty, but frequently fail to find the words to describe even mundane objects to which they are very familiar.

- **Global Aphasia**: Shown to be the most debilitating type of Aphasia, it is carried by patients who have suffered lesions in multiple parts of the brain, usually caused by strokes. Patients can suffer from all symptoms associated with Fluent and Non-Fluent Aphasia, and can also have severe difficulties related to speech impairment. In some cases, patients are almost entirely incapable of pronouncing words and/or understanding speech in general.

One can find many contributions from the fields of Psycholinguistics and Medicine that show practical investigations as to how Aphasia affects the reading skills of patients. The studies conducted with Fluent and Non-Fluent Aphasia patients described by Swinney et al. (1989) reveal very important information with respect to what types of words can represent obstacles for them. The aphasic subjects, as well as a control group of people without Aphasia, were asked to decide on which is the meaning of a word with respect to the sentence where they were extracted from. The results show that Fluent Aphasia patients and the

control group were, in general, not challenged by this task. In contrast, Non-Fluent Aphasia patients have shown to find difficult to identify the correct meaning of a word when such word does not represent its most frequently used meaning. This observation provides evidence that those affected specifically by Non-Fluent Aphasia may have problems understanding sentences with highly ambiguous words, or words that can be of multiple lexical categories, such as "*roll*", which can be both a verb (as in "*to roll on the ground*") and a noun (as a synonym for "*sandwich*" or "*streak*").

Studies about the speed of lexical activation in the task of lexical priming with aphasic patients have also provided valuable insight on the types of challenges faced by them. The task of lexical activation can be described as the process performed by the brain in accessing the semantic information related to a given word in order for one to decide on its meaning. The process of priming, on the other hand, can be described as a stimulation which is triggered at a given time in response to a previous stimulus. Lexical priming is an expression commonly used in literature to describe the process of deciding on the meaning of a word given a previously presented context.

Prather et al. (1992) describes a user study conducted with a single Non-Fluent Aphasia patient in order to find evidence of whether or not this type of Aphasia delays lexical activation in lexical priming. The subject was asked to decide on whether or not a given string of characters is a valid word of the English language after being exposed to an either related or unrelated reference valid word. In order to estimate the lexical activation speed of the subject, a series of ISI's (Inter-Stimulus Intervals) were tested. In this context, an ISI is the time for which the reference word stays on the screen before the subject is presented the target word for it to judge. The results show that larger ISI's allow the user to be more accurate, but that the average speed of lexical activation of the Non-Fluent Aphasia patient is slower than the one of healthy elderly subjects. Such results suggest that those affected by Non-Fluent Aphasia have problems comprehending sentences which are presented to them only temporarily over short spans of time, such as the subtitles of a movie. We are not aware of any LS systems that consider the task of simplifying subtitles for the aphasic.

In the user study of Prather et al. (1997), the speed of lexical activation of Non-Fluent and Fluent aphasic patients were compared. The experiment conducted with the subjects is nearly identical to the one of Prather et al. (1992), in which they are asked to decide whether or not a sequence of characters is a valid word of the English language after being presented an either related or unrelated word after an ISI. The results obtained are in accordance with the ones obtained by Prather et al. (1992): the Non-Fluent Aphasia patient presents slower speed of lexical activation in comparison to both healthy elderly subjects and the Fluent Aphasia patient. Such phenomenon has also been found by Milberg et al. (1987), who performed

almost identical experiments with Fluent and Non-Fluent aphasic subjects. The studies highlight the fact that each type of Aphasia is unique, and should be studied individually in order for LS systems to better suit their needs.

Aphasic patients were also the target audience in the work of Devlin (1999), a thesis focused on assessing and addressing the needs of the aphasic with respect to LS. His work provides a thorough analysis on the many varieties of Aphasia, as well as a survey on previous efforts regarding the usage of computational resources and methods in the creation of assistive software for the aphasic. Carroll et al. (1999) lists five techniques suggested by Lesser and Milroy (1993) on how to help the patients who suffer from linguistic-related conditions to recover from lost linguistic abilities:

1. **Reactivation**: Reiterating over how to perform the basic linguistic tasks which have become challenging.

2. **Reorganisation**: Conceiving and using new methods of teaching for affected patients to re-learn how to perform the linguistic tasks which have become challenging.

3. **Cognitive Relay Substitution**: Teaching affected patients new adapted strategies of performing challenging linguistic tasks.

4. **Substitution by Prostheses**: Incorporating prostheses such as electronic devices and computers as assistive tools to help patients perform challenging linguistic tasks.

5. **Functional Communication Strategies**: Refers to teaching patients personalised ways of performing challenging tasks, such as speaking and writing, so that they can participate in social and academic activities with more ease.

Considering the description of those techniques, one could argue that Lexical Simplification systems can be categorised as prostheses, which modify texts in order for patients to perform the task of reading more easily. The LS approach described by Carroll et al. (1999) selects the most frequent synonyms to replace the words of a given sentence.

Following the contributions of Devlin (1999), Carroll et al. (1998) and Carroll et al. (1999) present very similar LS strategies that also aim to assist the aphasic. They also extract synonyms of target words from Wordnet and select the one with the highest Kucera-Francis score to replace it. Devlin and Unthank (2006)'s approach is, to our knowledge, the first that incorporates Human-Computer Interaction resources in order to help patients to comprehend texts. They explore the concept of "memory jogging" which is the process of using suggestions of synonyms and related images to remind readers of the meaning of certain words and expressions.

Although simplifiers have greatly evolved throughout the years after these contributions have been published, there are not, to our knowledge, any more contributions that focus their attention on the necessities of aphasic patients with respect to LS. Modern contributions have focused mostly on improving the performance of simplifiers based on evaluation datasets annotated by English speakers of undisclosed background (Glavaš and Štajner, 2015; Horn et al., 2014).

We were not able to find any studies that investigate the needs of Anomic and Global aphasic patients in relation to Text Simplification. Considering how important the findings presented by contributions such as the ones of Prather et al. (1997), Prather et al. (1992) and Milberg et al. (1987) are, and the fact that there are very few records of LS systems that use those findings as motivation, we believe that there is still a high demand for contributions focused on helping the aphasic. Since there is already a wide range of simplifiers available, trying to understand what type of strategy better suits each of the main types of Aphasia would be a step towards effectively helping those affected.

2.1.2 Dyslexia

Dyslexia is a disorder commonly diagnosed during childhood, found to be caused by abnormalities in visual and auditory devices. Dyslexia can manifest itself at any stage of one's life, with causes ranging from blunt traumas to the head, to strokes and progressive diseases. As in the case of Aphasia, Dyslexia can be of different types, and patients can have multiple obstacles with respect to reading, writing and text comprehension. Some of the varieties of Dyslexia (Baddeley et al., 1982; Friedman and Hadley, 1992; Galaburda, 2006; Gitterman et al., 2012) are:

- **Deep Dyslexia**: Impairs the patient to accurately decide on the precise meaning of a word. Often, the patient will be able to read words, but will only be able to associate them with similar or related meanings.

- **Surface Dyslexia**: Partially impairs the patient's speaking skills. Frequently causes the patient to have difficulty in pronouncing words with unfamiliar phonetic structure.

- **Pure Dyslexia**: A type of Dyslexia which impairs the patient's reading skills, usually not affecting their speaking ability. The patient presents what is frequently referred to in literature as "letter-by-letter" reading, which causes reading to be considerably slower than that of a non-dyslexic.

- **Neglect Dyslexia**: Another variety of Dyslexia that affects the patient's reading capabilities. While reading, the patient ignores certain letters of words, which can sometimes hinder their capability of comprehending a word's meaning.
- **Attentional Dyslexia**: Characterised by rendering the patient confused when reading sentences or documents in which letters are closely grouped together due to lack of space. Patients often find it easier to comprehend the meaning of a sentence when reading one word at a time.

Each type of Dyslexia shows very distinct symptoms, and can be modelled with dedicated assistive tools. While patients with Attentional Dyslexia might benefit from a tool that rearranges the layout of a document in order to add more space between words, Pure Dyslexia patients might instead have problems while reading movie subtitles, and hence making them shorter could be of help.

In the context of Text Simplification, contributions can be found that evaluate how certain LS strategies influence the readability and understandability of sentences with respect to the needs of the dyslexic. Rello et al. (2013c) presents a study that investigates which forms of number representations are easier to comprehended dyslexic people. Their experiment investigates the following six hypotheses:

1. Numbers represented by digits, such as "10", are easier to read than numbers represented by words, such as "ten".
2. Numbers represented by digits are easier to comprehend than numbers represented by words.
3. Rounded numbers, such as "10", are easier to read than non-rounded numbers, such as "10.55".
4. Rounded numbers are easier to comprehend than non-rounded numbers.
5. Numbers represented by percentages, such as "10%", are easier to read than numbers represented by fractions, such as "1/10".
6. Numbers represented by percentages are easier to comprehend than numbers represented by fractions.

A total of 72 subjects participated in the experiment: 36 people with Dyslexia and 36 healthy control subjects. All subjects were asked to read texts containing one of the aforementioned number representations while wearing an eye-tracking device. Subjects were

also asked to answer questionnaires about the content of each text after reading them. In their work, they define readability as the speed with which a segment of text can be read. The use of eye-tracking allows for one to measure the fixation time spent by each subject while reading the excerpts, and hence judge how readable it is. The findings from their experiments are:

- Numbers represented by digits have shown to be significantly easier to read than written numbers for the dyslexic subjects. The readability measures for healthy subjects have shown not to be significantly different between the two representations.
- No statistical significance was found for either healthy or dyslexic subjects to support that numbers represented by digits are easier to comprehend than written numbers.
- No statistical significance was found for either healthy or dyslexic subjects to support that rounded numbers are easier to read than non-rounded numbers.
- No statistical significance was found for either healthy or dyslexic subjects to support that rounded numbers are easier to comprehend than non-rounded numbers.
- Percentages have shown to be significantly easier to read than fractions for the dyslexic subjects. The readability measures for healthy subjects have shown not to be significantly different between the two representations.
- No statistical significance was found for either healthy or dyslexic subjects to support that percentages are easier to comprehend than fractions.

Their results are evidence that the dyslexic can indeed be aided by LS in the context of numerical representations: using digits instead of words and percentages instead of fractions has proven to be an effective way of increasing text readability for patients. Another experiment that assesses how LS can help the dyslexic is described in (Rello et al., 2013b). Four hypothesis are tested in their experiment:

1. Sentences with a larger number of high frequency words offer increased readability to the dyslexic.
2. Sentences with a larger number of high frequency words offer increased understandability to the dyslexic.
3. Sentences with a larger number of short words offer increased readability to the dyslexic.

4. Sentences with a larger number of short words offer increased understandability to the dyslexic.

A group of 46 subjects participated in their experiment, 23 of which had Dyslexia and 23 were healthy control subjects. They were asked to read sets of texts that have been manually simplified in two ways: by replacing an arrange of selected words by their most frequent synonyms, and by their shortest synonyms. To estimate readability, an eye-tracking device was used, and in order to assess understandability, subjects were asked to answer questionnaires about the texts after reading them. The results revealed that:

- Dyslexic subjects showed considerably larger reading and fixation times than healthy subjects in general.
- The usage of more frequent synonyms has caused a significant reduction in the reading and fixation times of the dyslexic, but has no impact on reading times of the control group.
- Sentences with a larger number of high frequency words offer higher understandability to the dyslexic.
- The usage of shorter synonyms has caused a significant reduction in the reading times of the dyslexic, but had no impact on reading times of the control group.
- Sentences with a larger number of short words offer higher understandability to the dyslexic.
- The usage of more frequent synonyms has no impact on comprehension scores of both dyslexic and healthy subjects.
- The usage of shorter synonyms has shown a significant increase in the comprehension scores of the dyslexic.

While replacing words by their most frequent synonyms has been shown to increase the readability of a text, replacing words by shorter synonyms has been shown to significantly increase its comprehensibility. A small survey conducted by Rello et al. (2013b) also lists some contributions that have identified categories of words that represent a challenge to dyslexic people:

- Coltheart (1996) found that words which are not pronounced as their arrangement of syllables suggests, such as "*vase*" or "*butter*", can be challenging to the dyslexic.

- Ellis (1993) found that words which have very similar orthography, such as "*addition*" and "*audition*" can be confusing to the dyslexic.
- Patterson et al. (1985) found that homophonic words i.e. words that sound very similar but are written differently, such as "*weather*" and "*whether*", can also cause readability problems.
- Vega (2008) and Coltheart (1996) found that foreign, unfamiliar and made up words can also be obstacles for the dyslexic.

Rello et al. (2013a) presents another user study that identifies the needs of the dyslexic with respect to LS. They conduct an experiment to discover what type of text adaptation strategy better suits dyslexic people. Two adaptation strategies were tested:

1. **SubsBest**: Replaces complex words with more frequent synonyms. The system used to employ this strategy is LexSis, a simplifier introduced by Bott et al. (2012).
2. **ShowSyns**: Shows a list of available synonyms for the words that the users themselves annotate as challenging in the text. This system performs not simplification, but Text Elaboration, in which complex words and expressions are enhanced with complementary information that clarifies on their meaning.

A total of 96 subjects participated in the study, 47 of which were dyslexic. Subjects were asked to read four distinct versions of texts: its original unsimplified version, a version simplified by SubsBest, a version enhanced by ShowSyns, and a manually simplified gold-standard. An eye-tracking device was used to measure reading and fixation times. Subjects were also asked to answer multiple-choice questions about the content of the texts, and to judge in a five point Likert scale how easy each text was to read, understand and remember. The main conclusions drawn from the experiment are:

- The SubsBest strategy has been shown to not affect reading and fixation times of dyslexic subjects significantly, and actually showed lower comprehension scores in comparison to the ones obtained for the unsimplified version of the texts.
- The ShowSyns strategy has caused for a significant increase in comprehension scores for dyslexic subjects.
- Dyslexic subjects judged the texts read with the assistance of ShowSyns to be more easily readable, understandable and memorizable than both original, SubsBest and gold-standard versions. Nevertheless, non-dyslexic subjects disagreed, giving lower scores for SubsBest texts than to the original versions in all three aspects.

In summary, their experiments have revealed that even a modern LS system can compromise the understandability of a text. In contrast, a system that "suggests" synonyms to words that the user selects have shown very promising results, helping dyslexic people to comprehend texts.

2.1.3 Autism

Autism is a neurodevelopmental disorder that falls into the category of Autism Spectrum Disorders, and is characterised by an array of conditions that can cause an individual to suffer from several medical, behavioral and linguistic impairments. Among Autism Spectrum Disorders are also Asperger and Heller's syndromes, which provoke complications similar to the ones caused by Autism.

Baron-Cohen (2000) presents a comprehensive survey on the effects of Autism in the behavior of children. Most of the documented complications which are caused by the disorder do not affect the linguistic capabilities of the autistic directly, but can most definitely introduce challenges in reading comprehension and communication. Children that suffer from Autism often have problems distinguishing mental from physical actions (Stone et al., 2003). They have, for example, difficulty in understanding the difference in the degree of knowledge obtained by a person who thinks of a certain object, and a person who actually touches and/or examines it. Another impairment refers to understanding the function and capabilities of the brain: autistic children have more difficulty in distinguishing between processes that occur strictly in abstraction, such as thinking, dreaming and imagining, and processes administered by the brain which provoke physical responses, such as touching, walking and jumping. They are also known to have limited abstraction capabilities. When presented with objects of dual-identity, such as an apple-shaped candle, autistic children have been shown to be much less likely to correctly identify that the apple-shaped candle was not, in fact, an apple (Stone et al., 2003). There are various other examples mentioned by Baron-Cohen (2000) of complications caused by Autism:

- Autistic children often suffer from attention deficit, and struggle to concentrate on subjects or topics which are of no interest to them.

- Autistic patients can find challenging to infer someone's intent by observing their behavior. They can also struggle to identify deceit, sarcasm and humorous remarks.

- Autistic patients can find themselves incapable of diagnosing the emotional state of someone else, or predicting someone else's reaction to their actions.

- Autistic patients can consider only their opinion when making a decision, and hence sometimes disregard the impact of some of their actions in the emotional state of others.

Although not of linguistic nature, some of the aforementioned complications can also affect autistic patients' reading comprehension capabilities. When reading books with elaborate plots and complex character development, for example, an autistic person may find it challenging to comprehend the intent behind decisions made by characters, which could consequently confuse them.

Despite the interest of researchers from the Medicine and Psycholinguistics in understanding the problems caused by Autism, we were not able to find many examples of contributions that focus on either understanding or addressing the linguistic-related complications caused by them from a simplification perspective. Barbu et al. (2015) mention several problems caused by Autism Spectrum Disorders that directly affect the linguistic capabilities of patients:

- Although they are usually proficient in comprehending the meaning of words individually, they often cannot infer the meaning of an ambiguous word from its context.
- They often cannot link objects in sentences with pronouns and anaphoric references to them.
- Their incapability of linking pronouns and anaphoric references to objects becomes more severe as the syntactic complexity of a sentence grows.

Their simplification system, which is called Open Book, addresses such linguistic limitations by employing several strategies commonly used by successful simplifiers, such as:

- **Image Retrieval**: The user can select which words they do not comprehend and ask for an image search, which will retrieve images that illustrate them. Open Book acquires images by querying ImageNet (Deng et al., 2009), Wikipedia, Google and Bing.
- **Document Summarisation**: Open Book employs an algorithm similar to Google's Page Rank, in which the sentences of a given document are ranked according to how relevant they are to its overall content. The user can then retrieve a portion of the best ranking sentences as a summary of the document's content.
- **Document Topic Modeling**: By using a topic modelling approach called Latent Dirichlet Allocation (Blei et al., 2003), Open Book presents the user with the excerpts and expressions that best summarise the overall topic of a given document.

Their simplification approach was evaluated by 243 Autistic patients. They were asked to read the original and simplified versions of various documents, and then answer a questionnaire about the content of the document. They claim that subjects who read simplified versions of the documents have achieved noticeably higher comprehensibility scores than those who have read the documents in their original form. They do not, however, present any statistics that would allow one to quantify how effective their approach is in comparison to other strategies.

Yaneva et al. (2015) presents an eye-tracking study that also reveals much about how simplification could aid the autistic. By comparing the fixation times of subjects while reading various documents, they find that the autistic tend to fixate on photographs and images much more than the non-autistic. Their findings suggest that, while adding visual components that describe key components in a text can help the autistic in comprehending its meaning, polluting the text with ads and/or images that are not descriptive of any content in the text can hinder their capability of comprehending it. They also find that modifying texts by following the Plain English guidelines of Freyhoff et al. (1998) allow for the autistic to achieve noticeably higher comprehension.

One may notice that neither Barbu et al. (2015)'s or Yaneva et al. (2015)'s work present any type of LS strategy, which shows that finding effective LS approaches for the autistic is still an open problem. Given that those affected by Autism Spectrum Disorders find it challenging to understand ambiguous words, perhaps an effective way of performing Lexical Simplification would be by replacing them words with less ambiguous alternatives. Syntactic or Semantic Simplification could also aid the autistic: resolving anaphoric references could potentially increase reading comprehension.

2.1.4 Low-Literacy

Contrary to Aphasia, Dyslexia and Autism, Low-Literacy is not a medical, but rather a social condition that affects people which have not had access to quality education. The Instituto Paulo Montenegro[2] is a Brazilian institution that created the Functional Illiteracy Index (INAF), an initiative with the goal of measuring the levels of literacy in Brazil. The people involved in collecting such statistics visit the houses of families of different social classes, applying questionnaires to family members between the ages of 15 and 64 in order to assess their literacy levels. The INAF groups those who participated in the questionnaires in four categories of literacy:

[2]http://www.ipm.org.br

1. **Illiterate**: The individual is not capable of completing even basic tasks related to reading, but, in some cases, can understand familiar numeric expressions (such as telephone numbers).

2. **Rudimentary**: The person is capable of assessing basic information from short and familiar texts, read and write small numbers and perform basic tasks related to reading and writing, such as counting money to pay for bills and using a measuring tape.

3. **Basic**: People under this category are also described as "Functionally Literate". They are capable of comprehending medium-sized texts, can assess information from texts even when inference is required, read numbers in the magnitude of millions, can solve mathematical problems that involve a sequence of simple operations, and have a notion of proportionality. They are, however, incapable of solving problems involving an increased amount of steps, components or relations.

4. **Advanced**: People under this category are the ones whose literacy skills no longer impose limitations with respect to comprehending and interpreting the content of texts in general. They are capable of reading long texts and understanding the relations in them, can compare and evaluate pieces of information, can distinguish facts and opinions, and can perform inference and synthesis over a subject. They can also comprehend graphs, maps and tables, and solve more complex mathematical problems that involve planning, reasoning, percentages, proportions and measurements.

The PorSimples Project (Aluisio and Gasperin, 2010) is the biggest contribution in the field of Text Simplification for the Portuguese language. It aims to address the needs of those who are affected by low-literacy by providing tools that simplify texts in many different ways, such as through Summarisation, Syntactic Simplification, Lexical Simplification and Text Elaboration.

Watanabe et al. (2009) introduces Facilita, one of the tools created in the PorSimples project of which the goal is to perform real-time Syntactic and Lexical Simplification, as well as Text Elaboration to pages of the web. Facilita is a browser plug-in with an easy-to-use interface that allows for the user to customise the way texts are presented to them.

To our knowledge, there are no other contributions in literature that describe LS methods for low-literacy readers. The INAF statistics for the years of 2011 and 2012 indicate that only 26% percent of the Brazilian population is judged to have Advanced literacy skills, while 47% are Functionally Literate and the remaining 27% are judged to have either Rudimentary skills or to be completely illiterate. According a survey conducted by UNESCO (Huebler and Lu, 2012), the proportion of literate citizens can also be as low as 37% in some countries

in Africa and Asia. and These numbers show that low-literacy is still a serious problem in developing countries, and consequently highlight the dimension of the role that Text Simplification could still play in addressing this problem.

2.1.5 Second Language Learners

Second Language Learners are those who are at some stage, be it early or advanced, formally or informally, of acquiring a second language. Statistics provided by ETS (Austin et al., 2006) show that, throughout the years of 2004 and 2005, there were around 5.1 million English learners in the United States alone (10% of the country's population). They also report a growth in the number of English learners of over 700% in some states between the years 1994-1995 and 2004-2005. Such numbers have inspired many, specially in the field of Psycholinguistics and teaching, to investigate the impact of simplification methods and tools in the language acquisition process of second language learners.

One of the first contributions on the topic of second language learning was presented by Krasher (1985), who introduce five hypotheses regarding language acquisition:

- **The Acquisition-Learning Hypothesis**: States that there are two types of knowledge about a second language that are internalised in different ways: "acquired" and "learned". Acquired knowledge is described by Krasher (1985) as information which learners are not consciously aware of the rules, such as the appropriateness of a sentence in a certain context, while learned knowledge is described as information which learners can consciously grasp in a more formal and structured fashion.

- **The Natural Order Hypothesis**: States that the elements of a given language can only be learned in a certain order, which cannot be changed even under the influence of distinct teaching methods.

- **The Monitor Hypothesis**: States that the learned knowledge acts as a monitor which influences on the way that acquired knowledge is internalised. In order for said monitor to be effective, it must respect three conditions: it requires reasonable time, a focus on the form, and reliable knowledge of the rule.

- **The Input Hypothesis**: States that, in order for someone to be able to learn a second language, they must be iteratively presented with comprehensible, yet moderately challenging knowledge. This strategy is referred by the author as "i+1", where i is intended to represent the state of knowledge of a given learner, and $+1$ a moderate increment in the array of knowledge necessary for the learner to comprehend a given piece of

content. The hypothesis assumes the existence of an innate Language Acquisition Device (LAD) within the learner's mental faculties, which is activated whenever they are presented with comprehensible knowledge.

- **The Affective Filter Hypothesis**: Claims the existence of the "Affective Filter", which determines how much comprehensible knowledge can be presented to the LAD so that it acquires/learns new information about a second language. A high Affective Filter, which could be caused by low motivation, self-confidence or anxiety, would cause the learner to have limited or impaired knowledge acquisition.

The extent to which these hypotheses hold could greatly influence the decision of whether or not simplification could be used as a reliable tool for teaching, since it directly modifies the linguistic properties of the content to which second language learners are exposed. Throughout the years, such hypotheses have been frequently debated. In (Gass and Selinker, 2008), for example, is argued that the $i+1$ learning scheme proposed by Krasher does not provide any practical insight with respect to teaching or language acquisition, and is rather just a reductionist idealisation of the learning process. Burden (2006) mentions that there is increasing evidence that Krashen's Acquisition-Learning Hypothesis is flawed: contributions such as the ones of Larsen-Freeman (1991) and Pica (1994) show that the output produced by the learners' own speaking skills can also act as an input of acquired knowledge. In contrast, some authors claim that Krashen's hypotheses can in fact be observed in practical contexts. The work of Ellis (1994) presents evidence that there are certain dependencies between the nature of linguistic knowledge and the proficiency stage of a learner.

By using Krashen's hypotheses as an argument, some claim that simplification can compromise the second language learning process. Campbell (1987) states that simplified versions of texts can "water-down" the information inherent to the characters in a narrative, which could hinder the overall comprehension of certain types of content. A similar argument is made by Bacon and Finnemann (1990), who point out that simplification may cause an unwanted change in the cultural heritage inherent to certain writing styles, which could prevent the learner from overcoming cultural barriers during language acquisition. Yano et al. (1994) argues that a learner would not be able to progress in second language acquisition if only presented with knowledge they are already familiar with. The same argument is used by Honeyfield (1977) and Oh (2001).

In more practical setups, many have chosen not to debate over the theoretical aspects of language acquisition, but rather to investigate how the use of Text Simplification methods can assist second language learners. Some contributions have shown that syntactic simplification can actually lower the comprehensibility of texts. The work of Lotherington-Woloszyn

Original	Because he had to work at night to support his family, Paco often fell asleep in class.
Text Simplification	Paco had to make money for his family. Paco worked at night. He often went to sleep in class.
Text Elaboration	Paco had to work at night to earn money to support his family, so he often fell asleep in class next day during his teacher's lesson.

Table 2.1 Comparative example of Text Simplification and Text Elaboration, as described by Long and Ross (1993)

(1992), Ulijn and Strother (1990), Oh (2001) and Yano et al. (1994) suggest that syntactically modified sentences are often less cohesive than their original forms, and are rarely more easily comprehensible. As an alternative, Oh (2001), Parker and Chaudron (1987), Yano et al. (1994) and Long and Ross (1993) suggest Text Elaboration: a process which does not shorten or rearrange the content of sentences, but rather complements it in order to make it simpler. The example of Table 2.1, extracted from Long and Ross (1993), illustrates the difference between their rendition of Text Simplification and Text Elaboration.

Lexical Simplification, on the other hand, have shown much more promise as an assistive tool for second language learners. The results obtained by Tweissi (1998) reveal that LS can in fact increase the comprehensibility of texts significantly. Gardner and Hansen (2007) introduces a substantial compilation of contributions that highlight the potential of LS. Among those contributions are the ones of Nation (2001) and Hirsh et al. (1992), which present statistics obtained through experimentation about the relation between learned vocabulary and reading comprehension: English language learners need to be familiar with around 95% of a text's vocabulary in order to achieve basic comprehension, and with an even higher proportion for leisure (98%). Another very important argument in favor of LS has been outlined by Carver (1994) and Nation and Coady (1988), who show that, when a text's vocabulary is matched by the reader's, comprehension is often achieved even if the text is composed of several sentences of high syntactic complexity. In other words, their findings show that LS can, in some contexts, eliminate the need for Syntactic or Semantic Simplification altogether.

There are, however, those who believe LS can hinder language acquisition. Honeyfield (1977), for example, argues that LS causes a homogenisation of the content which a second language learner is exposed to, which can slow the process of acquiring new vocabulary. The experiments of Oh (2001), Yano et al. (1994) and Young (1999) present somewhat discouraging findings for LS. Their results show that there is no discernible increase in

comprehensibility of second language learners when different strategies of LS are applied in texts of varying subjects. Such arguments are nonetheless disputed by Gardner and Hansen (2007), who argue that, while Honeyfield (1977) has no practical evidence to support his claims, the methodology used in the studies conducted by Oh (2001), Yano et al. (1994) and Young (1999) impose an inherently strong bias against Text Simplification, given that they do not properly assess the opinion of second language learners about simplified texts. They also claim that the results obtained by them may be convoluted, since their simplification operations comprised not only lexical substitutions, but also syntactic operations such as reducing the sentence's length. Finally, Leow (1993) claims that the gains in comprehensibility commonly associated with the use of LS do not justify the costs of simplifying texts. This argument, however, takes simplification strictly as a manual task, which becomes thus an encouragement statement in favor of automatic approaches.

Given the arguments presented, we conclude that, although there is some evidence that Text Simplification can hinder or slow second language acquisition, the potential of LS systems to increase the comprehensibility of texts justifies investigating more effective approaches for the task.

2.1.6 Non-Native Speakers

A non-native speaker of a given language is anyone who speaks said language, but have learned a different first language earlier in their life. They can significantly vary in age, cultural background, proficiency and education level, and are certainly the largest and most diverse group among all the ones included in this survey. The audience of non-native speakers comprise not only second language learners, but also those who have already achieved fluent speaking status in a given language. Due to this diversity, researchers in the area of Text Simplification often prefer to conceive simplifiers for more restricted, homogenous groups, such as the dyslexic or aphasic.

We were not able to find many examples of user studies or LS approaches that address the needs of non-native speakers who, differently from second language learners, have to process language content without the purpose of learning the second language. The English Lexical Simplification task of SemEval 2012 (Specia et al., 2012) has allowed for the research community to submit and compare their strategies for LS. The data made available for training and testing was created by non-native English speakers with various backgrounds. Their findings show that the non natives' perception of word simplicity strongly correlates with word frequencies extracted from the Google 1T corpus (Evert, 2010)

But despite the scarcity of user studies with respect to Text Simplification specifically, there is a wide array of studies that investigate how the profile differences between native

and non-native English speakers affect the quality of their experiences as students and teachers. Leki (2001) presents a study that investigates how non-native speakers of the English language are perceived and treated in group projects. In their study, six non-native speakers of English had their academic experience documented throughout five years of their undergraduate courses. By interviewing the students about their performance in group projects, they were able to find that the group members who were native English speakers had low expectations with respect to their academic performance, and did not communicate as frequently with non-native speakers of English as they did with the other members of the group. By the end of the experiment, the non-native speakers of English showed a more pronounced contempt towards group projects than they did before it started.

Several user studies have also been conducted with native and non-native teachers of the English language and their students, most of which have been surveyed by Reves and Medgyes (1994). The results obtained suggest that students perceive significant distinctions between native (NET) and non-native English speaking teachers (NNET). It was found that students often perceive NETs as more skilled English speakers than NNETs. NNETs tend to use less usual sentence constructs during teaching, and often show to have a more limited vocabulary than the ones of NETs. In contrast, NNETs often offer more thorough explanations about the topics being taught, and are also more proficient in understanding and addressing the difficulties that students may encounter when learning something new about English.

Overall, most of the differences between NETs and NNETs are caused by divergences in English proficiency, which suggests that even skilled non-native English speakers may have somewhat limited fluency in comparison to native speakers. Such observation allows us to conclude that there is still the need to investigate what causes such drawbacks in their proficiency, specially with respect to limited vocabulary and speaking skills. We can also conclude that, since LS mainly focuses on simplifying the vocabulary required to comprehend a given text or sentence, it may also be an effective assistive tool for non-native speakers, specially those who are not fluent speakers of said non-native language.

2.2 Complex Word Identification

We now review every one of the steps in the typical LS pipeline. In the Complex Word Identification (CWI) step the goal is to select the words in a given sentence which should be simplified. Shardlow (2014b) illustrates an interesting example of this task: in the sentence "*The cat perched on the window*", the word "*perched*" is a clear candidate for simplification,

since one could argue that this is not a very common word, especially when compared to some of its synonyms, such as "*sat*" and "*rested*".

The following Sections discuss the advantages and limitations of six categories of strategies for CWI:

- Simplify Everything;
- Threshold-Based;
- Lexicon-Based;
- Implicit Complex Word Identification; and
- Machine Learning-Assisted.

2.2.1 Simplify Everything

Early LS approaches (Devlin and Tait, 1998) did not perform Complex Word Identification. Instead, they assumed that all words in a sentence could be simplified. This strategy has lost popularity over the years since, as demonstrated by Paetzold and Specia (2013), a simplifier without a CWI module might replace words which are already easy to understand by the target audience in question, and hence make the text even more difficult or less meaningful. In Devlin and Tait (1998)'s simplification approach, all words and phrases of a sentence are targets for simplification. Their results report that 16.60% of the simplified sentences had their grammatical structures compromised, while 44.50% of them had their meaning modified. (Paetzold, 2013; Paetzold and Specia, 2013) reported that manual inspection showed that many modifications made to the sentences were indeed performed over portions of text which did not need simplification, leading to the multiple cases of ungrammatical and/or incoherent substitutions.

2.2.2 Threshold-Based

Threshold-based approaches aim at searching for a threshold t over a given metric of simplicity M for a word w such that if $M(w) < t$ the word w can be more confidently categorised as a complex (or simple) word.

Keskisärkkä (2012) reports a study on the effect of using a word's length as a metric for CWI. Their LS approach simplifies sentences by replacing complex words with their most frequent synonym. Results show that increasing the word length decision threshold effectively decreases the number of errors performed by their approach, i.e. simplifying only

words with more than 7 letters in length proved to produce sentences with higher readability scores than simplifying all words in the sentence.

Word frequency has been a much more popular choice of metric for threshold-based approaches. Bott et al. (2012) describes an LS system for the Spanish language. By discarding substitutions for words in a sentence which appear in more than 1% of sentences in a large corpus, their method is able to outperform a baseline approach that does not discard any candidates. Considerable improvements in both meaning preservation and simplicity have been reported. Leroy et al. (2013) describe a similar approach that simplifies medical domain texts. They choose to simplify only words with an occurrence count of less than 15.377.914 times, which is the occurrence count of the 5000th most frequent word in the Google 1T (Evert, 2010). Human subjects reported a discernible reduction in the perceived reading difficulty in comparison to unsimplified sentences. Shardlow (2013a) learns the threshold that best separates complex and simple words from the CW corpus (Shardlow, 2013b). They use word frequencies from the SUBTLEX corpus (Brysbaert and New, 2009a) as a metric. Their threshold-based approach offers a noticeable improvement in performance over a baseline "simplify everything" approach.

Threshold-based approaches are intuitive in nature and easy to implement. However, the evaluation of a set documents in the Spanish language in (Bott et al., 2012) suggests that it is difficult to elaborate a single simplicity feature or metric that is capable of discerning between complex and simple words. Their analysis was performed over 40 manually simplified documents. When evaluating simpler synonyms of complex words, it was found that less than 70% of the simpler words were actually shorter in length than their complex equivalents. This means that simplifying only words which are more than t characters in length could lead to either ignoring certain complex words, or replacing simple words.

An analysis conducted by Shardlow (2014a) provides further evidence that threshold-based approaches are unreliable. The goal of their work is to find out the most frequent types of errors made by a baseline LS approach. Their LS approach is very similar to the one of Devlin and Tait (1998), and identifies simplifiable words by computing their Kucera-Francis coefficient: if it is lower than 5, the word is deemed complex and should be simplified. They define seven types of errors that can be made throughout the LS pipeline, two of which are CWI errors:

- **Type A**: A complex word is identified as simple.
- **Type B**: A simple word is identified as complex.

In order to find out how many mistakes are made by the system, a set of 115 sentences were simplified, and a manual evaluation performed by the author over the output produced

after each step of the LS pipeline. The findings show that a mistake was made in more than 65% of the complex word identification operations performed by the system, representing a total of 119 mistakes out of 183 operations. 99 of the 119 total mistakes were of Type B, which resulted in many simple words being unnecessarily (and incorrectly) replaced.

2.2.3 Lexicon-Based

To tackle the limitations of threshold-based approaches, some domain-specific LS systems have used a different approach for CWI. Their strategy consists in using a lexicon of complex words to identify simplifiable candidates: if a given word w is part of the lexicon of complex words L, then it should be simplified.

Deléger and Zweigenbaum (2009) present a method for building a lexicon of complex paraphrases in the medical domain. The method consists in automatically identifying aligned paraphrases in technical medical articles from the Web which have a corresponding summary written in lay terms. The approach uses a topic segmentation tool (Hearst, 1994) to identify pairs of segments likely to have aligned paraphrases, and then selects the pairs of segments which have a word co-occurrence vector cosine similarity higher than 0.33. The lexicon is hence composed by all segments extracted from documents written in technical terms.

When available, manually constructed lexicons can also be useful. The paraphrase extraction approach of Elhadad and Sutaria (2007) uses the lexicon provided by UMLS (Unified Medical Language System) (Bodenreider, 2004), a database of technical medical terms, to identify complex words and expressions. A more elaborate technique is used by Elhadad (2006): they consider simple all expressions from UMLS that can also be found in the Brown corpus. This filtering method explores the intuition that, if an expression that is commonly used in medical documents is found in a corpus of an unrelated domain, it is likely that such an expression is not complex. Through an experiment where college-level lay readers were asked whether or not a given medical expression was familiar, they found that abbreviations are extremely likely to be unfamiliar to readers. Based on these results, they also employed the heuristic that all abbreviations found in a sentence are considered complex to readers.

Kajiwara et al. (2013) present an automatic method for the extraction of paraphrases of complex words for children. The lexicon of simple words used is the *Basic Vocabulary to Learn*, a manually collected set of 5.404 words of the Japanese language that can help children communicate more efficiently.

Lexicon-based strategies can be very effective in practice. The success achieved by the FACILITA system, described in (Watanabe et al., 2009), is a good example of that. FACILITA is a tool designed to simplify Web pages, and it was part of the PorSimples project (Aluisio

and Gasperin, 2010), a simplification framework for low literacy readers of the Portuguese language. The lexicon used to identify simple words in the PorSimples framework is composed by words extracted from books for children, and a manually constructed list of very frequent words in news documents and a set of words judged "concrete" in (Janczura et al., 2007). The FACILITA tool proved to effectively assist low-literacy readers in comprehending texts of complex nature, such as news articles.

Lexicon-based approaches also have limitations, since manually creating large lexicons of complex or simple words can be prohibitively expensive. In addition, deciding which words should be present in the lexicon is a challenge since it is very unlikely that different individuals even within the same target audience will consider complex the same words.

2.2.4 Implicit Complex Word Identification

More recent approaches perform CWI not as an initial step in the process of simplification, but rather implicitly during other steps of the pipeline. They consider all words in a sentence to be targets for simplification, but during the simplification process they discard substitutions $(w_1 \rightarrow w_2)$ that, when applied, replace a word w_1 with a more complex alternative w_2.

Biran et al. (2011); Bott et al. (2012) define word simplicity metrics, and then discard candidate substitutions which are estimated to be more complex than the target word being simplified. The metric of Biran et al. (2011) is described in Equation 2.1, where $F(w,C)$ is the frequency of word w in corpus C, and $\|w\|$ the length of candidate w.

$$M(w) = \frac{F(w,\text{Complex})}{F(w,\text{Simple})} \times \|w\| \tag{2.1}$$

The metric used by Bott et al. (2012) is more elaborate and was devised to account for the simplification needs of the dyslexic. These readers find it difficult to comprehend words that are long and unfamiliar. For each candidate substitution of a given target complex word, they first calculate its length score, as illustrated in Equation 2.2.

$$score_{wl}(w) = \begin{cases} \sqrt{\|w\| - 4} & \textbf{if } \|w\| \geq 5 \\ 0 & \textbf{otherwise} \end{cases} \tag{2.2}$$

They then combine the length score with the word's frequency in a weighted linear function, as shown in Equation 2.3.

$$M(w) = \alpha_1 score_{wl}(w) + \alpha_2 \log(F(w,\text{Simple})) \tag{2.3}$$

A similar approach is used by the LS system of Glavaš and Štajner (2015), which only replaces a target word if it is less informative than the candidate substitution selected. Their informativeness measure is described in Equation 2.4, where V is the set of all words in a vocabulary.

$$M(w) = -\log \frac{F\left(w, \text{Google1T}\right) + 1}{\sum_{w' \in V} F\left(w', \text{Google1T}\right) + 1} \tag{2.4}$$

The denominator of Equation 2.4 is the same for any two words in vocabulary V, which means that their measure depends on no more than their word frequency. In other words, their LS system only simplifies a target word if it has a lower frequency than that of the candidate substitution selected.

A different strategy is employed by Horn et al. (2014), which simply adds a substitution $(w_i \rightarrow w_i)$ to the set of candidate substitutions of complex word w_i. In other words, the complex word becomes a candidate substitution for itself. If their system decides that the target word is the simplest alternative to replace itself, then the target word is not simplified.

These implicit approaches avoid the problem of trying to decide which words are inherently complex enough for simplification, and instead focus on the question of whether or not simpler substitutions can be found for a given word. Although these approaches can lead to a reduced number of ungrammatical and/or incoherent simplifications (Glavaš and Štajner, 2015), they may still lead to unnecessary replacements, given that they do not explicitly account for the LS needs of the target audience.

2.2.5 Machine Learning-Assisted

Very few LS strategies employ Machine Learning techniques in CWI. To our knowledge, the only approach published prior to our thesis is the one of Shardlow (2013a), which provides a comparison between a Support Vector Machine, a threshold-based strategy and the "simplify everything" approach. The threshold-based approach uses the frequency of the word in the SUBTLEX corpus (Brysbaert and New, 2009b) as a metric, which is composed of over six million sentences extracted from subtitles of assorted movies. The final threshold that separates complex from simple words was determined by 5-fold cross validation over all possible frequency values. The features used by the SVM approach are:

- Frequency in the SUBTLEX corpus;
- Number of films in SUBTLEX;
- Word length;

- Number of syllables;
- Number of senses; and
- Number of synonyms

The SVM model was trained and tested with the CW corpus (Shardlow, 2013a), which contains Wikipedia sentences with a single target complex word along with a simpler alternative. The results show that the SVM approach slightly outperforms the other strategies, but leads to the lowest recall. The scores obtained by the"simplify everything" approach, however, which does not perform any type of complex word identification at all, contradict the discussion provided in Section 2.2.1: it obtains over 84% F1, which is the highest score among the systems. We believe these results are due to the evaluation corpus used. In order to train their CWI systems, they require not only the complex word examples in the CW corpus, but also examples of simple words. In order to produce examples of simple words, they simply take one randomly selected word from a portion of the CW corpus' sentences.

We believe, however, that there is an issue with how the these negative examples are generated. If a Wikipedia editor judges that only one word should be simplified in a given sentence, it is implied that the remaining words were judged to be simple already. It is not clear why this simpler intuition was not used for the CW corpus. We believe that the results reported by Shardlow (2013a) do not capture the potential of Machine Learning methods in CWI, and that a more rigorous evaluation of such approaches must be performed for conclusions to be drawn.

2.3 Substitution Generation

Substitution Generation (SG) refers to the process of producing candidate substitutions for complex words. In the typical LS pipeline, an ideal SG strategy will be able find all words that can replace a given target complex word in all contexts in which it may appear. In the case of ambiguous words, the strategy would have to find candidate substitutions for all their possible senses, so that, at Substitution Selection time, the LS approach is able to select the one that fits the context of a sentence being simplified. This strategy would maximise the recall of candidate substitutions for complex words, and consequently allow for simpler output.

The biggest challenge in SG is to identify all possible words which can replace a complex word in the various contexts it may appear. An LS approach with a "perfect" SG strategy is also not guaranteed to produce sentences without errors: the Substitution Selection strategy

must be able to effectively select the candidate substitutions that fit a given context with respect to its grammatical constructions and meaning.

Existing SG approaches fall under one of two categories:

- Linguistic Database Querying; or
- Automatic Generation.

In the following Sections we discuss their advantages and limitations.

2.3.1 Linguistic Database Querying

When searching for candidate substitutions for complex words, using linguistic databases manually constructed by professionals is intuitively a very sensible approach: synonyms related words as given by human will certainly lead to substitutions that can replace complex words. However, building such resources can be very expensive and time consuming.

There are very few examples of LS approaches that address the task of creating a database of semantically related words. Ong et al. (2007) report the creation of a dictionary containing descriptions of complex medical terms. Similarly, Kandula et al. (2010) create a small database composed of paraphrases describing 150 complex medical expressions in lay terms.

Most work uses instead resources already consolidated to search for candidate substitutions. WordNet is the most frequently used database for LS. The earliest LS approaches in literature (Carroll et al., 1998, 1999; Devlin and Tait, 1998) used synonyms extracted from WordNet as candidate substitutions of complex words. Nunes et al. (2013); Sinha (2012) used not only the synonyms listed in WordNet, but also hypernyms and hyponyms. An empirical study conducted by Drndarević and Saggion (2012) provides evidence that related words other than just synonyms, such as hypernyms, hyponyms and meronyms, can also provide strong simpler candidates for complex words. Although WordNet has proved useful for Lexical Simplification, Shardlow (2014a) show that using only WordNet synonyms can limit the potential of LS, since WordNet does not cover all complex words in the English vocabulary, nor does it contain all candidates which can replace a complex word. In their experiment they evaluated the most frequent types of errors made by an LS approach similar to the one of Devlin and Tait (1998). The results showed that over 42% of the 164 errors made by their approach were caused by WordNet not having suitable simpler substitutions for complex words.

Combining multiple linguistic databases has shown to be a more reliable alternative for SG. Leroy et al. (2013) aim to simplify texts in the medical domain. As candidate substitutions, they use the relations provided by WordNet along with the ones provided

by UMLS (Unified Medical Language System) and Wiktionary[3]. UMLS provides a large ontology containing semantic relations between pairs of medical terms. This resource was also used in the LS approach of Chen et al. (2012), which replaces complex medical expressions with simpler equivalent lay terms in order to improve the performance of Statistical Machine Translation systems. The "define:" function of Google provides a dictionary definition of a given word, and has been used in (Elhadad, 2006) to provide definitions in lay terms for medical expressions.

Associating the querying of linguistic databases with an automatic approach has also shown promising results in (De Belder and Moens, 2010). They intersected the synonyms extracted from WordNet with a set of related words learned through a latent-variable language model. They found that the resulting reduced set had fewer spurious substitution candidates.

Since the content of WordNet may not offer enough coverage for languages other than English, researchers have also used other linguistic databases to find simpler candidate substitutes for complex words. The strategy of Bott et al. (2012) uses the Spanish OpenThesaurus[4] to find synonyms for complex words in Spanish. The same resource is used by Baeza-Yates et al. (2015) with a more sophisticated SG strategy. Instead of simply querying the database for synonyms of a given complex word, they first create a list of words for each meaning registered in the Spanish OpenThesaurus. Each list is composed of all words associated with such a meaning. They then enrich such lists by querying the Google 1T corpus for the words' frequencies, as well as for all 5-grams where they occur as the third word. Such a generation strategy allows a more effective choice for the most suitable replacement for a complex word in Substitution Selection.

SynLex[5] is a thesaurus for the Swedish language used by Keskisärkkä (2012) to find synonyms for complex words. The PorSimples project (Aluisio and Gasperin, 2010), which provides an LS approach for Brazilian Portuguese, uses sets of related words provided by the databases TeP 2.0 (Maziero et al., 2008) and PAPEL [6], which were created through a methodology that aims at maximising the coverage of synonyms and antonyms available.

2.3.2 Automatic Substitution Generation

As discussed in the previous Section, even though large linguistic databases can be of great help in gathering candidate substitutes for complex words, they are not always available or sufficient. Automatic SG approaches aim to extract candidates from other, less expensive

[3] http://en.wiktionary.org
[4] http://openoffice-es.sourceforge.net/thesaurus
[5] http://folkets-lexikon.csc.kth.se/cgi-bin/synlex
[6] http://www.linguateca.pt/PAPEL

resources, such as parallel corpora. Exploring this type of strategy is very important, since some languages do not have manually created linguistic resources.

The most straightforward automatic SG approach in the literature is that of Kajiwara et al. (2013). It takes advantage of dictionaries that do not include synonymy relations, but still provide word descriptions. The approach first queries a dictionary for a complex word's definition, and then uses a tagger to produce the Part-of-Speech (POS) tag of each word in it. It then extracts as candidate substitutions all words with the same POS tag as the target word. This strategy has been shown to be an effective alternative for the Japanese language.

Replacing complex words with equivalent paraphrases has also been shown to be a promising LS approach (Paetzold and Specia, 2013). Automatic paraphrase extraction is a task that has been addressed in many ways in recent years. It is not our goal to survey paraphrase extraction methods, but rather to discuss how these methods have been employed in the task of LS. Elhadad and Sutaria (2007) present a strategy to extract simpler paraphrases for medical expressions using articles related to medicine aligned at document level. Their approach uses contingency tables describing the contextual differences between expressions, along with statistical modelling to determine whether or not an expression found in a technical medical article is equivalent to a simpler expression found in an aligned document written in lay terms. They compare their set of paraphrases found with a gold-standard set of paraphrases produced by professionals in the area, achieving an F1 score of over 66%.

Deléger and Zweigenbaum (2009) also extract paraphrases from articles aligned at document level, but go a step further by performing topic segmentation to produce alignments at sentence level. Once aligned sentences have been produced, heuristics are applied in order to find nominalisations of complex expressions and paraphrases of neo-classical compounds (such as "*gastritis*").

The most widely used resource in automatic SG in the context of LS is the English Simple Wikipedia, which contains a subset of the articles from the original English Wikipedia edited so that more readers can understand them. Yatskar et al. (2010) describe one of the first automatic approaches that use Simple Wikipedia. It extracts paraphrases of complex terms from Simple Wikipedia edits marked with the "simplification" label. Their method has proven to extract many useful paraphrases, such as "*stands for*" $\rightarrow$ "*is the same as*", and "*indigenous*" $\rightarrow$ "*native*".

Tree transduction has also been used as a technique for the extraction of paraphrases for LS. Paetzold and Specia (2013) uses the tree transduction model by Cohn and Lapata (2009) to extract lexical simplifications from a corpus of parallel sentences taken from Wikipedia and Simple Wikipedia. A similar tree transduction approach is described in (Feblowitz and

Kauchak, 2013). They also use word alignment and syntactic trees to extract both lexical and syntactic simplifications.

Some approaches in the literature focus on finding only single word equivalences in parallel corpora, rather than looking for paraphrases of any size. An example is the method in (Kauchak and Barzilay, 2006). Using a series of syntactic and semantic filtering procedures, the approach extracts pairs of words with related meaning by comparing target texts in English produced by translation systems, and equivalent source reference texts produced by humans.

Extracting complex-to-simple word correspondences from a corpus of parallel sentences (Kauchak and Barzilay, 2006) has been proven to be effective as well, according to the results reported by Horn et al. (2014). They produce alignments over the parallel sentences, and then use various filtering techniques to extract the best alignments between complex words in Wikipedia sentences, and simpler words in Simple Wikipedia sentences. Specia (2010) takes TS as a complex-to-simple translation problem, and uses Statistical Machine Translation techniques to extract, among other types of simplification rules, lexical equivalences between words in Wikipedia and Simple Wikipedia. The biggest limitation in these approaches is that they require a parallel corpus of aligned sentences. These resources are however scarce and mostly available for the English language only.

To address this, Yatskar et al. (2010) uses Simple Wikipedia in a different way: instead of searching for word equivalences between aligned complex and simple sentences, they look for simplification operations in Simple Wikipedia edits. The effectiveness of their approach was not evaluated in practice, however. Biran et al. (2011) describe another approach that avoids the need for aligned parallel corpora. Instead of using aligned words as an initial set of candidate substitutions, they consider every pair of distinct words in the Wikipedia and Simple Wikipedia to be a possible simplification pair. They filter any pairs which are morphological variants of each other or that are not registered as either synonyms or hypernyms in WordNet. In an experiment, they hired humans to manually evaluate the quality of 65 simplified sentences produced by their approach with respect to grammaticality, meaning preservation and simplicity. The results show that 77.91% of the simplified sentences were grammatically correct, while 62.79% retained the meaning of the original sentence, and 75.58% were simpler than the original. Although they did not use an aligned parallel corpora to extract an initial set of candidate substitutions, they still require a parallel corpus and WordNet in order to filter word pairs that are not semantically related. However, their results have shown that associating automatic methods with the use of human created linguistic resources can be a very effective approach for SG.

In an effort to entirely avoid the need for manually created resources in SG, the strategy recently proposed by Glavaš and Štajner (2015) offers a completely unsupervised approach for the task. It resorts to word embedding models, which require only unannotated text corpora to be trained. In a word embeddings model, each word in the corpus vocabulary is represented by a unique vector containing a user-defined number of n real values. Given a trained word embeddings model and a complex word, the generator extracts as candidate substitutions the 10 words whose embeddings vector has the highest cosine similarity with the complex word, except for their morphological variants. The results obtained are comparable to the ones in (Horn et al., 2014).

2.4 Substitution Selection

The goal of Substitution Selection (SS) is to determine which of the candidates available for a given complex word fits the context of the sentence being simplified. This task is one of the most important in the LS pipeline, since it should prevent an LS system from performing lexical substitutions that alter the meaning and fluency of a complex sentence, and hence, in some cases, rendering it incomprehensible. SS is, nonetheless, the step with the fewest approaches reported in the literature.

The surveyed SS approaches can be divided in the following five categories:

- Select All Candidates;
- Explicit Sense Labeling;
- Implicit Sense Labeling;
- Part-of-Speech Tag Filtering; and
- Semantic Similarity Filtering.

2.4.1 Select All Candidates

Similarly to what has been observed for the task of CWI, early LS approaches do not address the task of SS at all, and instead choose to consider all possible substitutions of a given complex word as valid candidates for simplification (Carroll et al., 1998; Devlin and Tait, 1998).

The error analysis conducted by Shardlow (2014a) provides substantial insight on the impact of disregarding SS on the quality of simplifications produced. 115 sentences were

simplified by their approach, and it has been found that the absence of an SS strategy led to over 29% of simplifications to severely change the meaning of the original sentence. Since a single incoherent substitution can compromise the meaning of a sentence, the results obtained by Shardlow (2014a) imply that selecting all candidates could lead to almost a third of the simplified sentences produced by an LS approach to compromise the original meaning of the text.

2.4.2 Explicit Sense Labeling

LS approaches that explore Explicit Sense Labeling attempt to model SS as a Word Sense Disambiguation (WSD) task directly. This strategy uses classification methods to decide the sense label of an ambiguous target word in the sentence being simplified, and then selects as valid candidates those which have the same label. Such sense labels can be found in linguistic databases, such as WordNet.

Thomas and Anderson (2012) focus on the tasks of SG and SS. They evaluate three WSD and two lexicon reduction strategies for the creation of a reduced lexicon for a document. As sense labels, they use the "synsets" from WordNet. Evaluation is done by measuring the semantic distance between the content of the reduced lexicons produced over a given document and the lexicon of its manually simplified version. Their results seem promising, but they do not conduct experiments to investigate whether or not lexicon reduction helps in the task of LS.

Nunes et al. (2013) describe a full LS system that uses Explicit Sense Labeling. During SS, they select only synonyms found in WordNet which are linked to the meaning of a complex word. To determine a complex word's meaning, they employ the WSD approach of Navigli and Ponzetto (2010). In an experiment, they ask humans to determine whether or not a simplified sentence produced by their system is grammatical, and if it has the same meaning of the original sentence. They achieve 82% in meaning preservation, but grammaticality is compromised in 37% of the cases.

In an effort to address the lack of WSD resources for Spanish, Baeza-Yates et al. (2015) introduces a novel approach. In order to disambiguate a word to be simplified in a certain sentence, they first extract the 5-gram composed by the target word itself and two words to both its left and right. They then calculate the score of each of the word's sense in the Spanish OpenThesaurus by summing the frequency with which each synonym appears as the third token of the 5-gram in the Google 1T corpus. The synonyms pertaining to the sense with the highest frequency are then selected as suitable candidates. Although the results presented are promising, their strategy was not compared to other modern LS approaches.

However promising, Explicit Sense Labeling has several limitations in practice. Perhaps the most obvious one is the need for manually created sense/synonym databases, which are often limited to very few languages and quite expensive to produce. Due to the nature of WSD, Explicit Sense Labeling also hinders the capability of an LS approach to replace words with simpler multi-word paraphrases: while determining the sense of a single word is already challenging, determining the sense of a paraphrase is even more so.

2.4.3 Implicit Sense Labeling

An intuitive way to address some of the limitations of Explicit Sense Labeling is to use automatic methods to learn sense classes of complex words, instead of querying them from sense databases. De Belder and Moens (2010) describe, to our knowledge, the only example of LS approach that does so. They select as valid candidates only those substitutions which are grouped together by a latent variable language model trained over large corpora. Their experiments show that their LS approach effectively increases the readability of documents, but since they do not compare their approach with any other, it is difficult to draw conclusions with respect to whether or not their SS approach is more effective than state-of-the-art strategies.

Latent variable language models differ from standard n-gram language models by automatically learning latent classes which group words that appear in similar contexts. Such classes are often interpreted as "sense classes" that have a strong correlation with synonymy groups. Such language models, however, are difficult to produce: the complexity of the algorithms used to create them is often quadratic with respect to the number of classes which it is set to learn (Brown et al., 1992a).

2.4.4 Part-of-Speech Tag Filtering

Given the difficulty in determining the sense of complex words, some approaches use POS tags as surrogates for sense labels. The PorSimples framework (Aluisio and Gasperin, 2010) and the FACILITA system (Watanabe et al., 2009) are simplification systems for the Brazilian Portuguese language that select as valid candidates only those substitutions which have the same POS tag as the target complex word. The main motivation behind this strategy is the absence of reliable WSD resources such as sense-based databases for the Portuguese language.

This strategy has also been used in the context of tree transduction for simplification. Paetzold and Specia (2013) describe an LS approach which replaces not only words, but entire paraphrases with simpler alternatives by automatically learning lexico-syntactic substitution

rules through tree transduction. Each rule is represented by a pair $\left(T_{complex}, T_{simple}\right)$, where both $T_{complex}$ and T_{simple} are subtrees of constituency parses which represent similar content. To filter bad substitution rules, they discard all pairs $\left(T_{complex}, T_{simple}\right)$ in which the label of the root node in $T_{complex}$ is different from the one in T_{simple}. Their results reveal that using POS tags to filter bad paraphrases ensures grammaticality in over 83% of the cases.

Although POS tags may be able to successfully filter many inappropriate substitutions for complex words which are associated with multiple grammatical classes, such as "roll", they are not sufficient for meaning disambiguation. This can be a problem when an LS system has to decide on which substitutions are valid for highly ambiguous words, such as "pitch", which, aside from being both a verb and a noun, has 23 distinct meanings in WordNet. The meaning preservation results of Paetzold and Specia (2013) highlight this limitation: only 56.5% of the simplified sentences have the same meaning as their original version.

2.4.5 Semantic Similarity Filtering

Semantic Similarity Filtering consists in establishing a metric of the similarity between the meaning of a complex word in context and that of a substitution candidate, and then discarding all candidates which do not have enough meaning similarity with the complex word.

Biran et al. (2011) employs this SS strategy. After producing candidate substitutions for complex words, they create 10-token window co-occurrence word vectors $C\left(Sent\left(t\right)\right)$ and $C\left(c\right)$, where $C\left(Sent\left(t\right)\right)$ represents the semantic content of target word t in sentence $Sent$, and $C\left(c\right)$ the semantic content of candidate c in a large corpus. Finally, they discard candidates whose cosine distance between $C\left(Sent\left(t\right)\right)$ and $C\left(c\right)$ is lower than 0.1, a threshold achieved through experimentation.

The context-aware setup of this strategy is perhaps one of the most practical and promising approaches for SS with respect to meaning preservation. Biran et al. (2011) reports 75.86% of meaning preservation during simplification, which is the highest percentage reported in experiments with humans that we know of. Additionally, this approach does not rely on manually created linguistic databases, and hence can be applied to any language for which large corpora are available.

A similar, yet even simpler approach was proposed by Paetzold and Specia (2015). Instead of a co-occurrence model, they use a word embeddings model to determine the semantic similarity between a candidate and the context of a complex word being simplified. They rank all candidates according to the average cosine distance between each of them and the content words in the sentence, and then retrieve a proportion of the most similar

candidates. Although they do not provide with a human evaluation of their approach, their selector outperforms all other strategies evaluated.

2.5 Substitution Ranking

The last step in the typical LS pipeline is deciding which of the candidate substitutions that fit the context of a complex word is the simplest. In essence, the task of Substitution Ranking (SR) consists in, given the needs of a target audience, quantifying the simplicity of candidate substitutions, so that replacing a target complex word yields the simplest output possible.

Substitution Ranking has helped LS gain popularity in the NLP community. The English Lexical Simplification task of SemEval 2012 brought visibility to the task. The shared task attracted many participants and led to the introduction of various novel ranking approaches. More recent approaches have also used the corpus and metrics proposed back then (Horn et al., 2014; Shardlow, 2013b).

The following Sections discuss the merits and limitations of three categories of SR strategies:

- Frequency-Based;
- Simplicity Measures; and
- Machine Learning Assisted.

2.5.1 Frequency-Based

Although simple, frequency-based SR strategies are one of the most popular choices for LS systems, and can be quite effective. Approaches in this category explore the intuition that the more frequently a word is used, the more familiar it is to readers.

The most widely used frequency-based approach is Kucera-Francis (Rudell, 1993), a metric that determines the simplicity of a word given its frequency in the Brown corpus (Carroll et al., 1998; De Belder and Moens, 2010; Devlin and Tait, 1998; Shardlow, 2014a). Although popular, specially in early contributions, some studies show that the Kucera-Francis coefficient is not the most sensible approach for SR. Burgess and Livesay (1998), for example, argue that the frequencies taken from the Brown corpus, composed by roughly 1 million words, can be outperformed by raw frequencies from larger corpora. More recently, Brysbaert and New (2009b) have also shown that the origin of the corpora used to extract frequencies from can greatly influence ranking results: raw frequencies extracted from a corpus composed of movie subtitles have been shown to better capture word familiarity, and consequently

correlate better with word simplicity than the Kucera-Francis coefficient. In SR, subtitles have proven to be useful. The simplifier of Paetzold and Specia (2016i) uses a context-aware frequency-based SR approach. It first trains a language model over a corpus of subtitles of family movies, then ranks candidates according to their 5-gram frequency i.e. the candidate surrounded by two words to the left and right of the target word. Although simple, their ranker outperforms even Machine Learning-based supervised approaches.

In frequency-based ranking, most work use raw frequencies from very large corpora. Ligozat et al. (2012) use frequencies extracted from the Microsoft N-gram Services platform[7], which offers access to language models of up to 5-grams for the English language. Similarly, Leroy et al. (2013) and Baeza-Yates et al. (2015) use word frequency estimates from the Google 1T Corpus[8], composed by over one trillion words of the English language. Kauchak (2013) discusses how combining word frequencies obtained from simplified texts, such as articles from Simple Wikipedia, and frequencies obtained from unsimplified data, can improve on the performance of frequency-based rankers. A ranker which uses interpolated data between Wikipedia and Simple Wikipedia performs 23% better at the English Lexical Simplification task of SemEval 2012 than a ranker that uses only data from Simple Wikipedia.

Contributions that describe frequency-based rankers for languages other than English, or for specific text domains, also exist. (Keskisärkkä, 2012) is an example for the Swedish language: it uses the Swedish Parole database[9] as a source for word frequencies. Elhadad and Sutaria (2007), who target the simplification of medical content, rank lay expressions for technical medical terms according to their frequencies in a set of documents of the medical domain.

Search engines have also been often used as sources for word frequency estimates. The LS approach for the Portuguese language described in (Aluisio and Gasperin, 2010) ranks substitutions by their number of occurrences in pages retrieved through the Google API[10]. A similar ranking approach is presented in (Nunes et al., 2013): they use the Yahoo Search Engine API[11] to query for individual candidates and rank them according to the number of pages in which they appear. This strategy can be a very practical alternative for the task in online scenarios, since it discards the need for large language models trained over billions of words, and hence allows for lightweight simplifiers to be created.

In practice, frequency-based approaches have been shown to outperform more sophisticated ranking approaches quite often. In the results reported by Specia et al. (2012), a

[7] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx
[8] https://catalog.ldc.upenn.edu/LDC2006T13
[9] http://spraakdata.gu.se/parole/lexikon/swedish.parole.lexikon.html
[10] https://developers.google.com
[11] https://developer.yahoo.com

baseline that ranked candidate substitutions according to their raw frequencies in the Google 1T Corpus outperformed 9 out of 11 ranking approaches, and consequently placed 3rd overall.

There are, nonetheless, contrasting studies which show that word frequency is not sufficient to estimate word simplicity. Drndarević and Saggion (2012) demonstrate so by evaluating the characteristics of 40 manually produced correspondences between complex and simple words in the Spanish language. They found that the simple words had a higher usage frequency than their complex counterparts in only 54.76% of the cases.

2.5.2 Simplicity Measures

An alternative to address the limitations of frequency-based ranking strategies are metrics that incorporate multiple features to represent the simplicity of a word. The metric introduced by Biran et al. (2011), for example, considers a word's frequency and length to determine its complexity. Their metric is shown in Equation 2.5, where $Comp(c)$ is the corpus complexity of candidate c, and $\|c\|$, its lexical complexity.

$$M(c) = Comp(c) * \|c\| \tag{2.5}$$

In Equation 2.5, the corpus complexity is computed as illustrated in Equations 2.6, where $F(c,C)$ is the raw frequency of candidate c in corpus C. The "Complex" and "Simple" corpora required by the metric must contain text of complex and simple nature, respectively.

$$Comp(c) = \frac{F(c,\text{Complex})}{F(c,\text{Simple})} \tag{2.6}$$

A similar metric is used by Sinha (2012). It combines a candidate substitute's length, number of senses in WordNet, and frequency of occurrence in various corpora to determine a word's simplicity. Following the same notation used in Equation 2.6, the metric is computed as illustrated in Equation 2.7, where $S_{wn}(c)$ is the number of senses of c in WordNet.

$$\begin{aligned} M(c) = & F(c,\text{Simple Wiki}) + \\ & F(c,\text{Speech}) + \\ & F(c,\text{Google1T}) + \\ & S_{wn}(c) + \\ & \frac{1}{\|c\|} \end{aligned} \tag{2.7}$$

The $F(c, \text{Simple Wiki})$ and $F(c, \text{Google1T})$ are word frequencies extracted from Simple Wikipedia (Kauchak and Barzilay, 2006) and the Google 1T corpus (Evert, 2010). The "Speech" corpus in $F(c, \text{Speech})$ is a proprietary compilation of written dialogue content. The metric obtained the 2nd highest ranking scores in the English Lexical Simplification task of SemEval 2012.

Bott et al. (2012) describe a word simplicity measure for the Spanish language. They also consider a word's length and usage frequency in their measure, but go a step further, as described in Equation 2.8, to include two weighted scores, where α_1 and α_2 are adjustable weights.

$$M(c) = \alpha_1 score_{wl}(c) + \alpha_2 score_{freq}(c) \tag{2.8}$$

In order to estimate weights α_1 and α_2, Bott et al. (2012) resort to a heuristic search that maximises the score of the measure over a set of manually created lexical simplifications. In Equation 2.8, the values of $score_{wl}(c)$ and $score_{freq}(c)$ are calculated as illustrated in Equations 2.9 and 2.10, where $F(c, \text{Simple})$ is computed over the Spanish Simplext Corpus (Bott and Saggion, 2011).

$$score_{wl}(c) = \begin{cases} \sqrt{\|c\| - 4} & \textbf{if } \|c\| \geq 5 \\ 0 & \textbf{otherwise} \end{cases} \tag{2.9}$$

$$score_{freq}(c) = \log(F(c, \text{Simple})) \tag{2.10}$$

The motivation behind $score_{wl}(w_c)$ comes from the observation that, in their set of manually crafted lexical simplifications, the complex words in Spanish have on average four characters more than their simple counterparts.

Simplicity measures can also be more sophisticated, and incorporate the relation between the candidate and the context of the complex word to be simplified. Kajiwara et al. (2013), for example, represent simplicity as the weighted sum of five metrics that consider various relations between substitution candidate and the sentence to be simplified. Their simplicity measure was designed for Japanese, and can be described as illustrated in Equation 2.11, where *Sense* is the WordNet distance between the senses of candidate c and target t, *Cooc* the co-occurrence sum of the words in sentence S and candidate c, *Log* the normalised

co-occurrence sum between c and S, $Trigram$ the frequency sum of all trigrams surrounding c when replacing target word t in S, and Sim the distributional similarity between c and t.

$$\begin{aligned} M(S,t,c) = & \alpha_1 F_{corpus}(c) + \\ & \alpha_2 Sense(c,t) + \\ & \alpha_3 Cooc(c,S) + \\ & \alpha_4 Log(c,S) + \\ & \alpha_5 Trigram(c,S) + \\ & \alpha_6 Sim(c,t) \end{aligned} \tag{2.11}$$

We refer the reader to (Kajiwara et al., 2013) for more details on how $Cooc$, Log, $Trigram$ and Sim are calculated. A manual evaluation of the output produced by this LS approach reveals promising results, but the SR strategy itself has not yet been compared to others.

Although the aforementioned metrics have managed to achieve promising results in SR, only very recently efforts have been made to automatically combine various of these metrics without the help of annotated data. Glavaš and Štajner (2015) introduced the first ranking strategy that attempts to do so. Instead of manually crafting a single metric and ranking candidates accordingly, they combine the rankings obtained by several metrics into one. They resort to a very simple strategy: rank averaging. Their approach first produces the various rankings resulting from the use of several metrics, such as n-gram frequencies and semantic similarity. It then produces the score of each word by averaging all its rankings. Finally the words are ranked according to their score: the lower the average rank, the simpler it is. In their experiments, the approach outperformed all other systems in the dataset of the Lexical Simplification task of SemEval 2012.

2.5.3 Machine Learning Assisted

Given the effectiveness of modern Machine Learning techniques for various language processing tasks, they have also been adopted for SR. The most successful SR approach submitted to SemEval 2012 (Jauhar and Specia, 2012) uses Support Vector Machines (Joachims, 2006) along with the combination of various ranking functions to order candidates by simplicity. They represent ranking functions as described in Equation 2.12, and employ them to determine the ranking of a candidate c. In Equation 2.12, r_i is a standalone ranking function that

determines the rank of candidate c according to a certain metric, with respect to target word t in sentence S.

$$M(S,t,c) = \sum_{i=1}^{m} \frac{1}{r_i(S,t,c)} \tag{2.12}$$

The approach considers various ranking metrics, both context-aware and context-unaware, such as n-gram language model probabilities and psycholinguistic word properties. In order to optimise performance, they create various combinations of M ranking functions, use their resulting ranking values as features, and then train an SVM ranker over a very small set of training instances (300).

SVM ranking approaches have also been used in other work. Horn et al. (2014) trains an SVM ranker with various features: word frequency and n-gram probabilities from various corpora. Their LS approach outperformed two strong baselines in their evaluation, highlighting the potential of Machine Learning in SR. The main advantage of these strategies is that they are able to learn a model from data produced by human annotators, which could facilitate the process of adapting an SR approach to different target audiences.

2.6 Conclusions

We have presented a survey on Lexical Simplification which addresses the task as a series of steps: Complex Word Identification, Substitution Generation, Substitution Selection and Substitution Ranking. We have also presented a survey on user studies conducted with the various target audiences addressed in Text Simplification.

For Complex Word Identification, it is clear that supervised approaches that use modern Machine Learning techniques are more effective than threshold and lexicon-based alternatives. Exploring the use of other Machine Learning techniques beyond standard classification techniques, such as Neural Networks, could yield even more effective approaches for the task.

Virtually all Substitution Generation strategies are context-independent i.e. they do not take into account the context of a word being simplified when producing candidate substitutions. We believe that generators that account not only for the grammatical forms of complex words, but also their many possible senses should be explored. An intelligent generator capable of accounting for a word's grammatical class and sense while producing candidate substitutions would prevent incoherent and/or ungrammatical substitutions, and could also discard the need for a dedicated Substitution Selection step. Another limitation of generators lie in the fact that none of those surveyed are able to produce multi-word

expressions for complex words. Pairing paraphrasing strategies with single word candidate generators could yield better results.

An alternative to including a dedicated Substitution Selection step in simplifiers, is to use word embedding models. The proved success in selecting candidates through semantic similarity filtering, as well as the remarkable recent advances in word embedding modelling, lead us to believe that future selectors would benefit from exploiting them.

There is also a noticeable absence of user studies and analyses focusing on understanding concepts such as grammaticality and meaning preservation from a Text Simplification standpoint. We believe that dedicated studies, such as the one described in the Sections to follow, which aim at understanding what makes a candidate suitable to replace a complex word, would benefit research in the field.

By analysing the intricacies of various Substitution Ranking approaches, it became clear that strategies which automatically combine various features in supervised fashion are more efficient than word frequency and hand-crafted metrics, which require an extensive engineering process. By simply gathering human annotations from members of a given target audience, one could use these supervised strategies to learn a model of their needs.

Given the considerations made, we can state that there is much left to be explored in Lexical Simplification, particularly with respect to further understanding the needs of the various audiences commonly targeted by simplifiers. We believe that the Text Simplification research community would greatly benefit from the results of user studies such as those introduced by Rello et al. (2013b) and Rello et al. (2013c), which aim to learn more about specific readability and comprehensibility challenges caused by certain language impairments. These studies would not only serve as a foundation for the creation of new simplification strategies, but also result in new useful datasets and resources to be exploited in future research.

It is also safe to say that there are still many interesting opportunities to be explored when it comes to incorporating modern Machine Learning models and strategies in LS. Deep Neural Networks, for example, have been successfully used to push the state-of-the-art in various challenging Natural Language Processing tasks, such as Machine Translation (Zou et al., 2013), Sentiment Analysis (Glorot et al., 2011), Semantic Similarity (Yih and Qazvinian, 2012) and Question Answering (Iyyer et al., 2014). To this day, there are no examples in literature of simplifiers that employ, for example, deep Recurrent Neural Network architectures. These architectures could be used in strategies that jointly model the entire Lexical Simplification process, and hence disregard the need to engineer each step individually.

Chapter 3

User Studies with Non-Native Speakers of English

Researchers in Text Simplification often resort to user studies to find more information about reading challenges that a certain target audience may face. In Chapter 2 we present several contributions on the findings of user studies conducted with various target audiences. The user studies presented by Rello et al. (2013b), Rello et al. (2013a) and Rello et al. (2013c), for example, are a pioneer effort that have allowed us to understand much more about how texts should be simplified for dyslexic readers. As discussed in Section 2.1.6, however, there are no similar user studies that focus on the simplification needs of non-native speakers of English of a given language, the focus of this thesis.

Since non-native English speakers comprise one of the largest and most diverse groups that could benefit from LS, we believe that much could be gained from user studies that focus on how LS can help them overcome comprehension barriers. In this Section, we present the findings of three user studies that we conduct: Complex Word Identification, Substitution Selection and Substitution Ranking. For each one of them, we describe the data and methodology used, as well as the profile of annotators who participated. We conduct analyses on the annotations produced, and use the resulting datasets to conduct benchmarks and explore new approaches to LS.

3.1 Complex Word Identification

An interesting first step in developing an LS strategy for non-native speakers of English would be to understand the characteristics inherent to words they find complex. Such information could then allow one to conceive a Complex Word Identification strategy that predicts which

words will challenge them in reading. This information could be gathered from a user study, such as the ones of Rello et al. (2013a,b,c), conducted with dyslexic subjects. A user study on CWI would also address another gap in literature: the absence of reliable datasets for the task.

As discussed in Section 2.2, the CW corpus (Shardlow, 2013b) is the first dataset introduced for CWI. Although a relevant contribution, this dataset contains only 731 instances extracted automatically from the Simple English Wikipedia edits, which raises concerns about its reliability and applicability. Another limitation of the CW corpus is that there is no way of knowing who were the annotators who produced the simplifications therein, which in turn does not allow us to use this data to achieve our goal of outlining the needs of non-native English speakers.

In an effort to provide better resources and new insights for CWI in the simplification of texts for non-native English speakers, we have conducted a user study with 400 volunteers of varying age, native language, education level and English proficiency level. In addition to providing a better understanding on the characteristics of complex words, our user study aimed to create a sizable dataset for CWI, as well as to evaluate the applicability of various resources commonly used in the creation of LS approaches. In the study, volunteers were asked to judge whether or not they could understand the meaning of each word in a given sentence. The following Sections describe the user study in more detail.

3.1.1 Data Sources

We selected 9,200 sentences to be judged, after filtering out cases with spurious characters, HTML or CSS markup, or outside the 20-40 word-length range. These sentences were taken from three sources:

- **CW Corpus** (Shardlow, 2013b): Composed of 731 sentences from the Simple English Wikipedia in which exactly one target word has been simplified by editors from the standard English Wikipedia. 231 sentences that conformed to our criteria were extracted.

- **LexMTurk Corpus** (Horn et al., 2014): Composed of 500 sentences from the Simple English Wikipedia containing one target word simplified from the standard English Wikipedia. 269 sentences that conformed to our criteria were extracted.

- **Simple Wikipedia** (Kauchak, 2013): Composed of 167,689 sentences from the Simple English Wikipedia, each aligned to an equivalent sentence in the standard English

Wikipedia. We selected a set of 8,700 sentences from the Simple Wikipedia version that conformed to our criteria and were aligned to an identical sentence in Wikipedia.

The sentences in other LS datasets in literature did not conform to our criteria, so we decided not to include them.

3.1.2 Annotation Process

400 non-native speakers of English speakers of English participated in the user study, all university students or staff. Volunteers provided anonymous information about their native language, age, education level and English proficiency level according to CEFR (Common European Framework of Reference for Languages). They were asked to judge whether or not they could understand the meaning of each content word (nouns, verbs, adjectives and adverbs, as tagged by Freeling (Padró and Stanilovsky, 2012)) in a set of sentences, each of which was judged independently. Volunteers were instructed to annotate all words that they could not understand individually, even if they could comprehend the meaning of the sentence as a whole.

A set of 200 sentences was split into 20 subsets of 10 sentences, and each subset was annotated by a total of 20 volunteers. The remaining 9,000 sentences were split into 300 subsets of 30 sentences, each of which was annotated by a single volunteer.

3.1.3 Dataset Analysis

A total of 35,958 distinct words were annotated (158,624 in total), of which 3,854 distinct words (6,388 total) were deemed as complex by at least one annotator. In the following sections, we discuss details of the data collected.

Nature of Complex Words

We collected statistics that highlight the differences between simple words and those deemed complex by the annotators. The features we have calculated are 15:

- **Morphological:** Word length and number of syllables, according to Morph Adorner (Burns, 2013).

- **Semantic:** Number of senses, synonyms, hypernyms and hyponyms, according to WordNet (Fellbaum, 1998).

- **Lexical:** N-gram language model log-probabilities from the SubIMDB (Paetzold and Specia, 2016i), Subtlex (Brysbaert and New, 2009b) and Simple Wikipedia (Kauchak, 2013) corpora.

We trained a 3-gram language model using SRILM (Stolcke, 2002) from all aforementioned corpora in order to estimate n-gram probabilities. We choose these features and resources because, to our knowledge, they are the most widely used word complexity indicators in Text Simplification literature.

Table 3.4 shows the average feature values and standard deviations for all complex and simple words in the part of the dataset annotated by 20 volunteers. We define as complex any word which has been judged so by at least $n \in \{1,5,10\}$ annotators. The $[i,j]$ indicators present in the n-gram features of Table 3.4 refer to the number of tokens to the left (i) and right (j) of the words that was considered. Consequently, $[0,0]$ refer to single-word frequencies. The column succeeding feature values indicate whether there was (•) or not (○) a statistically significant difference between complex and simple words ($p<0.01$), given the results of an F-test[1].

	n=1			**n=5**			**n=10**		
Feature	Complex	Simple	p	Complex	Simple	p	Complex	Simple	p
Length	6.7 ± 2	6.6 ± 2	○	7.5 ± 2	5.9 ± 2	○	7.1 ± 2	6.1 ± 2	○
Syllables	2.1 ± 1	2.2 ± 1	○	2.3 ± 1	1.8 ± 1	○	2.2 ± 1	1.7 ± 1	○
Senses	6.6 ± 8	8.2 ± 8	•	2.1 ± 2	9.1 ± 9	•	1.1 ± 1	8.8 ± 9	•
Synonyms	16 ± 21	20 ± 21	•	5.3 ± 6	22 ± 23	•	2.3 ± 3	22 ± 22	•
Hypernyms	4.8 ± 6	5.5 ± 6	•	1.7 ± 2	6.1 ± 8	•	0.9 ± 1	5.9 ± 7	•
Hyponyms	24 ± 48	32.3 ± 64	•	4.0 ± 13	36 ± 52	•	0.8 ± 2	32 ± 52	•
Subimdb(0,0)	-5.3 ± 1	-4.8 ± 1	•	-6.5 ± 1	-4.6 ± 1	•	-6.6 ± 1	-4.5 ± 1	•
Subtlex(0,0)	-10 ± 21	-5.0 ± 5	•	-31 ± 41	-4.6 ± 1	•	-51 ± 46	-4.4 ± 1	•
Simple(0,0)	-5.9 ± 10	-4.3 ± 1	•	-11 ± 22	-4.2 ± 1	•	-8.4 ± 14	-4.2 ± 1	○
Subimdb(1,1)	-11 ± 3	-10 ± 3	•	-12 ± 3	-11 ± 3	•	-13 ± 3	-9.7 ± 3	•
Subtlex(1,1)	-19 ± 28	-13 ± 18	•	-40 ± 45	-16 ± 23	•	-59 ± 52	-13 ± 21	•
Simple(1,1)	-11 ± 18	-8.7 ± 9	•	-16 ± 26	-7.9 ± 2	•	-10 ± 15	-8.1 ± 2	○

Table 3.1 Average and standard deviation of features of words deemed complex or simple by at least n annotators. The $[i,j]$ indicators refer to the number of tokens to the left (i) and right (j) considered by n-grams. The p columns' values indicate the presence (•) or not (○) of a statistically significant difference.

The results shed some light on word complexity for a non-native English speaker. They show that, unlike semantic and lexical features, length and number of syllables have little to do with complexity. Although it has been found that shorter words do promote understandability

[1]The F-test is employed in these types of context to test the hypothesis that the variances from two population samples (complex and simple words, in this context) are equal under the assumption that the null hypothesis is true.

for the Dyslexic (Rello et al., 2013b), our findings contradict the belief that longer words tend to be more difficult to understand regardless of the audience being targeted (Biran et al., 2011; Shardlow, 2013a).

When it comes to semantic properties, it can be noticed that, while both average and standard deviation values for complex words decrease as the number of complex judgments increase, the same does not happen for simple words. This phenomenon suggests a relationship between ambiguity and complexity, where complex words are more likely to be unambiguous. This finding is in line with those of Shardlow (2013a), who successfully modelled word complexity by exploring the hypothesis that complex words tend to be less ambiguous.

Interestingly, a somewhat similar relationship can be observed between lexical properties and word simplicity: while the n-gram probabilities of complex words are high in both average and variance across all scenarios, the average probabilities of simple words are much lower, and vary much less.

Table 3.2 illustrates statistics on the relation between grammatical classes and word complexity. It shows the proportion between the number of occurrences of a grammatical class in which the word was deemed complex by at least one annotator, and its total number of occurrences in the dataset.

Class	Complex	Total	Proportion
Nouns	**2,826**	**66,600**	4.24%
Verbs	913	28,255	3.23%
Adjectives	756	15,837	4.77%
Adverbs	198	7,504	2.64%
Prepositions	9	28,029	0.03%
Foreign	122	390	**31.28%**
Other	14	42,974	0.00%

Table 3.2 Number of complex occurrences of grammatical classes

We found that nouns compose the largest portion of complex words in the dataset, having more complex occurrences than all other word classes combined. Foreign words, however, have shown to contain the largest proportion of complex occurrences in the dataset. A foreign word is considered complex almost a third of the time. Such words are often of Latin origin, and are commonly used in academic articles ("*et. al.*", "*priori*", "*posteriori*", etc).

Profile of Annotators

Annotators are speakers of 45 languages. The most predominant languages were Portuguese (15.3%), Chinese (13%) and Spanish (11.3%). Annotators are between 18 and 66 years old

(average 28.2). 63.7% of the volunteers were Postgraduate students, 32.3% Undergraduate, and 4% were in High School. 36.8% claimed to have Advanced (C2) English proficiency skills, 37.7% Pre-Advanced (C1), 16.6% Upper-Intermediate (B2), 6.4% Intermediate (B1), 2% Pre-Intermediate (A2) and 0.5% Elementary (A1).

By inspecting the data, we found interesting correlations between the number of complex words annotated and volunteers' age or English proficiency level. Figures 3.1 and 3.2 illustrate the average number of words deemed complex and their standard deviation values with respect to annotators in 10-year age bands and proficiency levels, respectively.

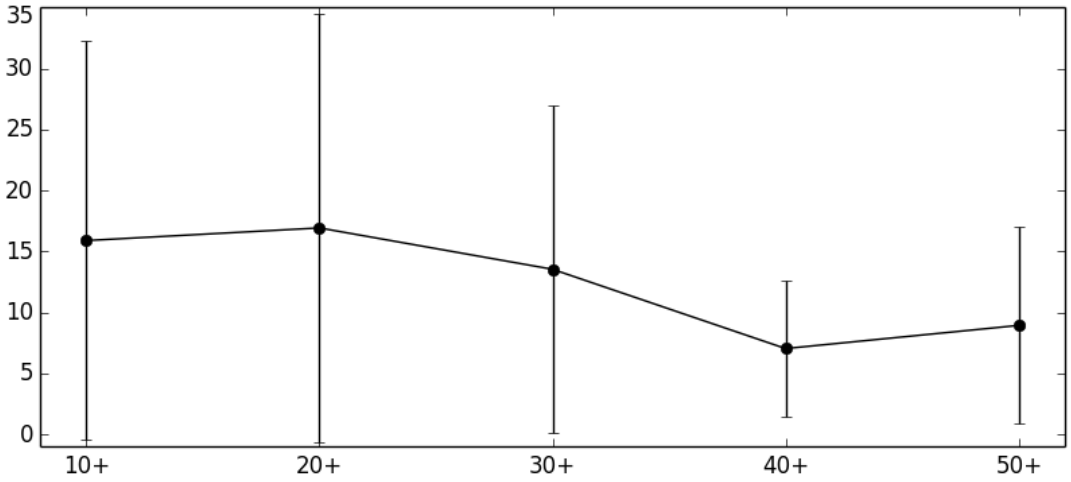

Fig. 3.1 Age bands over complex words. The horizontal axis represents age bands, and the vertical axis represents the average number of words deemed complex by the annotators in each band.

Both graphs show that, although the average number of complex words drops as age and proficiency level increase, the variance within each group is very high, suggesting that such groups may not be significantly distinct from each other. By performing F-tests with $p=0.05$, we found a significant difference between the band of 40+ years of age and the bands of 10+, 20+ and 30+ years of age, which suggests that one's English knowledge peaks at such age band. We also found significant differences between almost all English proficiency levels above A2, except between B2 and C1, which happen to have identical descriptions in the London School Level Scale[2]. We did not find significant differences among education levels.

Inspecting the annotations, we have also found that the speakers of certain languages are sometimes challenged by words which, in most cases, are not considered complex by native speakers of any other languages. Table 3.3 illustrates the words with the highest percentage

[2]http://www.londonschool.com/level-scale

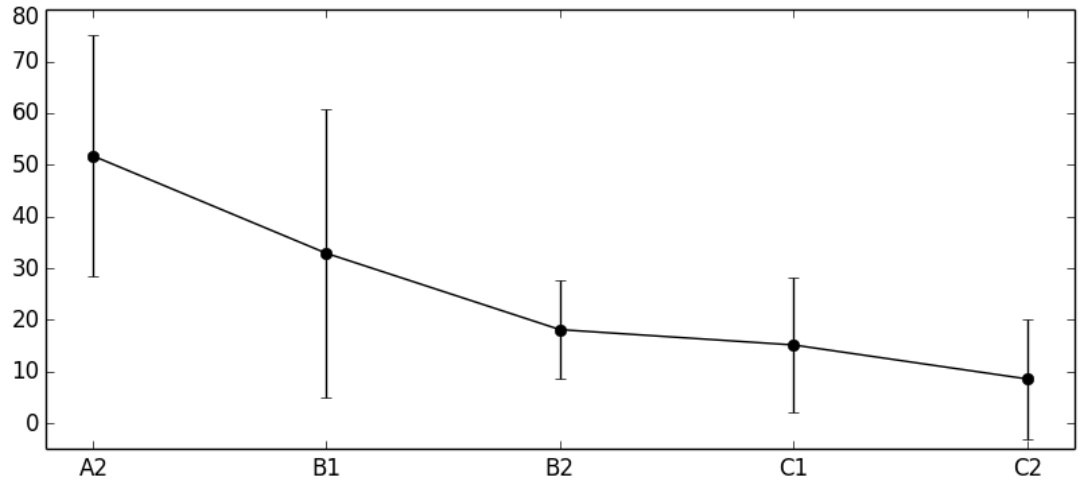

Fig. 3.2 Proficiency levels over complex words. The horizontal axis represents English proficiency levels, and the vertical axis represents the average number of words deemed complex by the annotators in each proficiency level.

of variance (Brysbaert and New, 2009b) between the number of times that they were deemed complex by the speakers of a specific native language, and the rest of the annotators.

Language	1	2	3	4	5	6
Arabic	fur	juvenile	apprenticed	city	serologic	link
Chinese	canton	inscribed	opium	referendum	thorax	contaminants
French	sewerage	subsequent	warships	escudo	dye	ridges
German	escape	strong	early	city	escudo	iconoclastic
Portuguese	rather	hurricane	undergo	southern	ruler	crude
Spanish	bailed	cryptanalysis	plaque	debris	demise	perm

Table 3.3 Words with highest percentage of complexity variance per native language. Indexes in the first row indicate the words' percentage of variance rank, from highest to lowest.

Analysis of Data Sources

We collected several statistics about how well the perception of word complexity adopted by Wikipedia editors correlate with that of our non-native speakers of English. Evaluating the data, we found that the words deemed complex by Wikipedia editors were marked as complex by our annotators in only 0.8% of the CW instances, and 19.7% of the LexMTurk instances. In contrast, 51.9% of the edited words in the CW corpus and 40.8% of those in the LexMTurk corpus were deemed complex by at least one of our annotators. As for the remaining Simple Wikipedia instances, we found that at least one word in 27.3% of the

instances was deemed complex by an annotator, which shows that the simplified version of Wikipedia may still be challenging to non-native speakers of English.

We also inspected these and other datasets for the purposes of LS. In addition to the aforementioned CW and LexMTurk corpora, we took the dataset used in the English Lexical Simplification task of SemEval 2012, composed of 2,010 instances total, and the LS evaluation dataset introduced by De Belder and Moens (2012b), composed by 430 instances. Each instance in all these datasets contains a sentence and a target complex word. Table 3.4 shows the number of distinct target words included in each dataset, how many of them appear in at least one of our 9,200 sentences, and the proportion of the latter that was deemed as complex by at least one of our annotators.

	Total	Appear in 9,200	Complex
CW	272	260	34.6%
LexMTurk	454	420	33.3%
SemEval	410	342	26.0%
De Belder	80	59	20.3%

Table 3.4 Results of dataset analysis for target words

Table 3.5 shows the same statistics as the ones illustrated in Table 3.4, except that they pertain not to the target words present in the corpora, but rather the substitutions suggested for them in Simple Wikipedia. Note that since the CW corpus does not provide substitutions for the target word of each instance, it is omitted in Table 3.5.

	Total	Appear in 9,200	Complex
LexMTurk	3,943	2,342	22.7%
SemEval	3,216	1,732	22.5%
De Belder	664	427	21.8%

Table 3.5 Results of dataset analysis for substitutions

While the statistics in Table 3.4 show that most target words in the previous corpora are not considered to be complex by non-native English speakers, the ones in Table 3.5 reveal that, on average, one in five substitutions listed in those corpora are considered to be complex as well. Such figures highlight the fact that the aforementioned resources are not appropriate for the training or evaluation of CWI or LS approaches targeting non-native speakers of English, since they do not necessarily capture their needs with respect to simplification.

Agreement Analysis

We have calculated Kappa's pairwise inter-annotator agreement coefficient (Carletta, 1996) for all pairs of annotators who were presented with sentences from the overlapping portion of the data. The values average to 0.616 ± 0.05, which is much higher than the agreement scores obtained by previous authors in similar tasks. While the annotators in the Lexical Substitution tasks of SemEval 2007 (McCarthy and Navigli, 2007) and 2010 (Mihalcea et al., 2010) obtain an agreement of 0.277, the annotators in the English Lexical Simplification task of 2012 obtain an agreement of 0.398.

More impressive yet is the agreement within certain classes of annotators. Although the agreement for annotators with the same proficiency level is lower (0.575 ± 0.07), the agreement within education levels and age bands are noticeably higher, reaching 0.638 ± 0.08 and 0.671 ± 0.08, respectively. Yet the highest agreement belongs to annotators with the same native language: 0.718 ± 0.1.

3.1.4 Benchmarking Automatic CWI Methods

In what follows, we discuss and compare several categories of approaches for automatic CWI using the data produced in our user study.

Datasets

We created two training sets for the task: *optimistic* and *conservative*. Both contain all 200 sentences which were annotated by 20 distinct annotators. In the *optimistic* training set a word receives label 1 (complex) if at least one volunteer annotated it as complex, whereas in the *conservative* training set a word is only deemed complex if judged so by at least five distinct annotators. The test set is comprised of all the remaining sentences which have been annotated by only one volunteer ($9,000$).

Each instance is composed by a sentence, an annotated word, and its label. The training set contains $2,237$ instances and the test set, $88,221$ instances. Using this setup, we are able to create a realistic scenario for our performance assessment: the CWI systems must decide on the individual needs of a person based on the overall needs of the target audience which they are part of.

Approaches

The approaches compared in this experiment can be divided in four categories:

- **Lexicon-Based (LB)**: A word is deemed complex only if it does not appear in a lexicon of simple words. We consider lexicons from three distinct sources: Ogden's Basic English (Ogden, 1968), the Simple Wikipedia (Kauchak, 2013), and a portion of the SubIMDB corpus (Paetzold, 2015b), which is composed of $4,351$ subtitles of movies and series for children.

- **Threshold-Based (TB)**: Given a certain feature, we find the threshold t that best separates complex from simple words. We consider the words' length, number of syllables, senses, synonyms, hypernyms and hyponyms in WordNet, and frequency in four distinct corpora: Wikipedia and Simple Wikipedia (Kauchak, 2013), SUBTLEX (Brysbaert and New, 2009b) and SubIMDB (Paetzold, 2015b). To find t, an exhaustive search was performed on the training set over $10,000$ equally distant values in the interval between the minimum and maximum value of each feature.

- **Machine-Learning Assisted (MA)**: Using various Machine Learning techniques, we train classifiers to decide whether or not a given word is complex. They are trained over the previously described training sets, and use 69 distinct features, which can be grouped into:

  - **Binary**: If a target word is part of a certain vocabulary, then it receives label 1, otherwise, 0. We extract vocabularies from Simple Wikipedia (Kauchak, 2013), Ogden's Basic English (Ogden, 1968) and SubIMDB Paetzold (2015b).
  - **Lexical**: Includes word length, number of syllables, number of senses, synonyms, hypernyms and hyponyms in WordNet (Fellbaum, 1998), and language model probability in Wikipedia (Kauchak and Barzilay, 2006), Simple Wikipedia and SubIMDB.
  - **Collocational**: Language model probabilities of all n-gram combinations with windows $w < 3$ to the left and right of the target complex word in Wikipedia, SUBTLEX, Simple Wikipedia and SubIMDB.
  - **Nominal**: Includes the word itself, its POS tag, both word and POS tag n-gram combinations with windows $w < 3$ to the left and right, and the word's language model backoff behavior (Uhrik and Ward, 1997) according to Simple Wikipedia.

  With the help of scikit-learn (Pedregosa et al., 2011), we learn models using the following seven techniques:

  - Support Vector Machines (SVM)

  - Passive Aggressive Learning (PA)
  - Perceptron
  - Decision Trees
  - Ada Boosting
  - Gradient Boosting
  - Random Forests

  With the help of Keras[3], we train a Multi-Layer Perceptron voter as well. Its architecture, including number and size of hidden-layers, is entirely decided through 5-fold cross-validation over the training set. The aforementioned models use as input all binary, lexical and collocational features. Finally, we also train a Conditional Random Fields model (CRF) using CRFSuite (Okazaki, 2007). It uses as input all nominal features described in the previous Section. The hyper-parameters of all Machine-Learning-assisted voters, including kernels, loss functions and learning rates, are optimised through 5-fold cross-validation.

- **Baselines (BA)**: We include the "All Complex" and "All Simple" baselines, which predict that all words are complex or simple, respectively.

Metrics

To assess the systems' performance, we choose to diverge from the typical F-score, which is the harmonic mean between Precision and Recall. Even though the F-score is arguably the most frequently used evaluation metric to compare the performance of classifiers, we feel that, as far as the relationship between Complex Word Identification and Lexical Simplification are concerned, it does not accurately capture the effectiveness of a strategy for the task.

To motivate our decision, we must first outline the characteristics of a good lexical simplifier. In order to be both effective and reliable, it must accomplish two things simultaneously:

1. To make a text as simple as possible.
2. Not to make any replacements that compromise the sentences' grammaticality and/or meaning.

In order to help a simplifier achieve these goals, a complex word identifier must consequently:

[3] http://keras.io

1. Avoid labeling complex words as simple, and hence impede them from being simplified.

2. Avoid labeling simple words as complex, and hence allow for unnecessary, possibly erroneous simplifications.

3. To capture as many complex words as possible, and hence maximise the simplicity of a sentence.

Now that we have outlined what the ideal complex word identifier must do, we can translate these objectives into typical evaluation expressions used in the context of classification problems. In this context, "positive" and "negative" decisions refer to labeling words as complex and simple, respectively.

While objectives number 1 and 2 state that the identifier must minimise the number of false negatives and false positives, item 3 states that it must maximise the proportion of true positives. The proficiency of a classifier in achieving these goals is measured by **Accuracy** and **Recall**, respectively. In order to balance these two metrics, we have conceived the **G-score**, which measures the harmonic mean between Accuracy and Recall. For completeness, we also include Precision and F-scores in our benchmark.

Results

The results for the *optimistic* and *conservative* settings are shown in Tables 3.6 and 3.7, respectively. They show that Decision Trees and Random Forests achieve the highest G and F-scores in the *optimistic* setup, but perform much less impressively in the *conservative* setup. This phenomenon repeats itself with much more intensity, however, for Conditional Random Fields (CRF), Adaptive Boosting, Passive Aggressive Learning (PA) and Neural Networks: while they offer very competitive G and F-scores in the *optimistic* setup, they drop to values close or even equal to zero in the *conservative* setup.

We believe that the cause for this is the reduced amount of complex words present in the *conservative* setup. While the *optimistic* training set contains 706 complex words, the *conservative* setup contains 117, which barely more than 5% of instances. This has caused these classifiers to acquire a strong bias towards simple words, which have consequently led them to neglect complex words almost entirely.

3.1.5 The Complex Word Identification Task of SemEval 2016

Using the datasets produced in our user study, we have proposed and organised the first run of the Complex Word Identification task as part of SemEval 2016. The participants had the

Cat.	Approach	A	P	R	F	G
LB	Ogden's	0.248	0.056	0.947	0.105	0.393
LB	Simple Wikipedia	0.953	0.241	0.002	0.003	0.003
LB	SubIMDB	0.913	0.217	0.332	0.262	0.487
TB	Length	0.332	0.057	0.852	0.107	0.478
TB	Senses	0.436	0.068	0.861	0.125	0.579
TB	Synonyms	0.436	0.067	0.853	0.124	0.577
TB	Hypernyms	0.572	0.076	0.728	0.137	0.641
TB	Hyponyms	0.384	0.065	0.906	0.121	0.539
TB	Freq:Simple Wiki	0.513	0.081	0.902	0.148	0.654
TB	Freq:Wikipedia	0.536	0.084	0.901	0.154	0.672
TB	Freq:SubIMDB	0.445	0.072	0.912	0.133	0.598
TB	Freq:SUBTLEX	0.492	0.077	0.896	0.142	0.636
MA	PA	0.852	0.171	0.562	0.262	0.677
MA	SVM	0.715	0.061	0.357	0.105	0.476
MA	CRF	0.534	0.076	0.808	0.140	0.643
MA	Perceptron	0.648	0.057	0.423	0.101	0.512
MA	Decision Trees	0.805	0.158	0.733	0.260	**0.767**
MA	Adaptive Boosting	0.799	0.153	0.728	0.253	0.762
MA	Gradient Boosting	0.802	0.147	0.672	0.241	0.731
MA	Random Forests	0.826	0.170	0.698	**0.273**	0.756
MA	Neural Networks	0.691	0.105	0.741	0.183	0.715
BA	All Complex	0.047	0.047	1.000	0.089	0.089
BA	All Simple	0.953	0.000	0.000	0.000	0.000

Table 3.6 Results for the *optimistic* training set

Cat.	Approach	A	P	R	F	G
LB	Ogden's	0.248	0.056	0.947	0.105	0.393
LB	Simple Wikipedia	0.953	0.241	0.002	0.003	0.003
LB	SubIMDB	0.913	0.217	0.332	0.262	0.487
TB	Length	0.591	0.067	0.595	0.120	0.593
TB	Senses	0.731	0.107	0.648	0.184	0.687
TB	Synonyms	0.754	0.106	0.574	0.179	0.652
TB	Hypernyms	0.660	0.084	0.635	0.149	0.647
TB	Hyponyms	0.687	0.092	0.644	0.162	0.665
TB	Freq:Simple Wiki	0.848	0.186	0.665	0.290	0.745
TB	Freq:Wikipedia	0.834	0.175	0.689	0.279	**0.754**
TB	Freq:SubIMDB	0.743	0.125	0.752	0.215	0.747
TB	Freq:SUBTLEX	0.842	0.176	0.648	0.277	0.732
MA	PA	0.953	0.364	0.001	0.002	0.002
MA	SVM	0.709	0.111	0.747	0.194	0.728
MA	SGD	0.730	0.111	0.681	0.191	0.705
MA	CRF	0.946	0.167	0.037	0.061	0.071
MA	Decision Trees	0.931	0.282	0.308	**0.294**	0.462
MA	Adaptive Boosting	0.953	0.504	0.044	0.081	0.084
MA	Gradient Boosting	0.595	0.079	0.716	0.142	0.650
MA	Random Forests	0.943	0.348	0.245	0.287	0.389
MA	Neural Networks	0.953	0.000	0.000	0.000	0.000
BA	All Complex	0.047	0.047	1.000	0.089	0.089
BA	All Simple	0.953	0.000	0.000	0.000	0.000

Table 3.7 Results for the *conservative* training set

goal of creating systems that, given a sentence and a target word within it, predict whether or not the target word would be deemed complex by a non-native English speaker. Participants were allowed to use any complementary external resources they desired.

Datasets and Evaluation

We have provided two training datasets for the task: **joint** and **decomposed**. Both of them contain all instances which were annotated by 20 non-native speakers of English. The **joint** dataset is identical to the one used in the *optimistic* setup in the benchmarking of Section 3.1.4. It contains a single label for each instance, which is 1 if at least one of the 20 annotators have deemed it complex, and 0 otherwise. In contrast, the **decomposed** dataset contains one label for each of the 20 annotators, which is 1 if they have judged it to be complex, and 0 otherwise. The test set is also the same used in the setup of our benchmarking.

For evaluation, we use the same metrics from our benchmarking, which are Accuracy, Precision, Recall, F-score and G-score.

SV000gg: CWI with System Voting

In order to participate in the task and compete with the various systems proposed by other authors, we have conceived the SV000gg systems, which employ two voting techniques.

Our strategy explores the hypothesis that the popular saying "*two heads are better than one*" apply to classification problems. We believe that combining the "opinion" of various approaches to a given task can yield better results than if they are applied individually. This idea is not new, and have been thoroughly explored in several ways.

Strategies that combine multiple Machine Learning classifiers are often referred to as Ensemble methods. Such methods range from very simple approaches, such as Hard Voting, in which labels are determined based on how many times they were predicted by the classifiers considered, to very elaborate approaches, such as Random Forests (Breiman, 2001) and Gradient Boosting (Friedman, 2001).

Our strategy consists on a variant of Soft Voting, in which the class of a given instance is determined through Equation 3.1.

$$c_f = \arg\max_c f(c) = \sum_{s \in S} T(s,c) \tag{3.1}$$

In traditional Soft Voting, c_f is the selected class, c is one of the possible classes in a classification problem, S the collection of systems considered, and T a confidence estimate i.e. a function that expresses how sure system s is that c is the correct class. Its goal is to increment Hard Voting by incorporating the systems' classification proficiency in the

decision process, hopefully leading to a more reliable way of exploiting their strengths and weaknesses.

Although sensible in principle, Soft Voting might not able to effectively combine systems if they do not have a reasonably uniform way of determining the confidence of their predictions. The presence of over-optimistic or over-pessimistic systems may skew the results severely, and hence make the resulting classifier have a worse performance than that of the best system among the ones considered. Another obvious limitation of traditional Soft Voting is that it cannot include systems which simply do not have an established way of determining the confidence of their prediction. Lexicon-Based approaches, for example, tend to be very effective in certain contexts, but can only produce binary confidence estimates: if the word is in the vocabulary, then they are 100% sure the word is simple, if not, they are 100% sure the word is complex.

To address these limitations, we propose a different ensemble approach: Performance-Oriented Soft Voting. Instead of using the systems' summed confidence to predict a label, it uses their performance score over a certain validation dataset. Formally, we decompose function T from Equation 3.1 into the two functions illustrated in Equation 3.2.

$$c_f = \arg\max_c f(c) = \sum_{s \in S} P(s,d) * D(s,c) \tag{3.2}$$

In Equation 3.2, P represents the score of system s over a certain dataset d given a certain performance metric, such as Precision, Recall, F1, Accuracy, etc. Function D, on the other hand, outputs value 1 if system s has predicted c for the classification problem in question, and 0 otherwise.

This setup works under the assumption that the systems' performance under a validation dataset is a reliable surrogate for confidence predictions, and allows for any type of systems to be combined, be they homogeneous in their way of predicting classes or not.

To train our systems for the CWI task of SemEval 2016, we combine the n systems with the best G-score among the Lexicon-Based, Threshold-Based and Machine-Learning-Assisted approaches benchmarked in the *optimistic* setup of Section 3.1.4. We select n through 5-fold cross-validation over the *joint* dataset.

Submitted Systems

Each team was allowed to submit at most two distinct systems. In total, 40 systems were submitted. The 20 teams that have submitted systems are:

AI-KU Introduces two SVM classifiers trained with a Radial Basis Function over the joint dataset. While one of their systems use as features the word embeddings of the target word itself and its sub-strings (native), the other uses the embeddings of the surrounding words as well (native1) (Kuru, 2016).

AKTSKI Presents two SVM classifiers: one that weights labels according to the annotators' judgements (wsys), and another that does not (svmbasic). Their systems use various semantic and morphological features, and were trained over the joint dataset.

Amrita-CEN Introduces two SVM classifiers trained over the joint dataset. While one of them uses word embeddings as well as various semantic and morphological features (w2vecSim), the other also includes POS tag information (w2vecSimPos) (sp et al., 2016).

BHASHA Presents two systems: an SVM (SVM) and a Decision Tree (DECISIONTREE) classifier. The instances in the dataset are first pre-processed, then classified according to various lexical and morphological features. Finally, the results are pos-processed with hand-crafted rules. Both systems are trained over the joint dataset (Choubey and Pateria, 2016).

ClacEDLK Uses Random Forests to train two classifiers over the joint dataset with semantic, morphological, lexical and psycholinguistic features. While one classifier uses a class-assignment threshold of 0.5 (RandomForest-0.5), the other uses a threshold of 0.6 (RandomForest-0.6) (Davoodi and Kosseim, 2016).

CoastalCPH Introduces a Neural Networks and a Logistic Regression system. Their Neural Networks system (NeuralNet) is trained over the joint dataset, and uses two hidden layers leading to a single activation node. Their Logistic Regression system (Concatenation) is trained over the decomposed dataset. Both systems use the same set of features, which include word frequency measures and word embedding values (Bingel et al., 2016).

GARUDA Presents two approaches: a hybrid model (HSVM&DT) and an SVM classifier ensemble (SVMPP). HSVM&DT obtains predictions from various SVM models, which are then validated by Decision Tree classifiers trained specifically to judge whether the predictions are correct. The validated predictions are then combined into a final label. SVMPP trains a single SVM classifier for each of the 20 annotators of the decomposed dataset, then uses a weighted average to combine their predictions (Choubey and Pateria, 2016).

HMC Performs CWI through a Decision and a Regression Tree, both with a maximum depth of four. During training, their systems deem complex those words which were judged so by at least 25% (DecisionTree25) and 5% (RegressionTree05) of the first 19 annotators in the decomposed dataset. Their systems are then tuned based on the judgment of the 20th annotator (Quijada and Medero, 2016).

IIIT Resorts to Nearest Centroid Classification to perform CWI. While one of their classifiers uses the Manhattan distance during training (NCC), the other uses the Euclidean distance (NCC2). As features, they use semantic and morphological features. Their systems are trained over the joint dataset (Palakurthi and Mamidi, 2016).

JUNLP Presents a Random Forest (RandomForest) and a Naive Bayes (NaiveBayes) classifier trained over the joint dataset. Among the semantic, lexicon-based and morphological features used are also the words' POS tag and Named Entity information (Mukherjee et al., 2016).

LTG Uses a very simple setup of Decision Trees trained over the decomposed dataset. Both of their systems learn a threshold based on the number of complex judgments in the decomposed dataset. While one of them learns only one threshold (System1), the other combines various (System2) (Malmasi et al., 2016).

MACSAAR Introduces a Random Forest (RFC) and an SVM (NNC) classifier. Instead of the usual, they instead use Zipfian features, such as the percentile ranking of the target word, and character n-gram features, such as the probability sum of all character n-grams in the sentence. For training, they use the joint dataset (Zampieri et al., 2016).

MAZA Employs Ensemble methods over the joint dataset. They train a context-independent system (A) that uses various word frequency features, and a context-aware system (B) that also includes frequency of the previous and following words (Malmasi and Zampieri, 2016).

PLUJAGH Presents two threshold-based approaches to CWI. Their first system (SEWDF) judges a word to be complex if its frequency in Simple Wikipedia is lower than 147. Their other system learns the frequency threshold from the joint dataset that maximises the F-Score (SEWDFF) (Wróbel, 2016).

Pomona Uses threshold-based bagged classifiers with bootstrap re-sampling. The thresholds of their classifiers are determined through brute-force over the target words' frequencies in a given corpus. They use bag sizes of 10 re-samplings selected through 10-fold cross validation, repeated 20 times. The corpora used are Wikipedia (NormalBag) and the Google Web Corpus (GoogleBag). Their systems are trained over the joint dataset (Kauchak, 2016).

Sensible System that combines Recurrent Neural Networks and Ensemble Methods. Their Neural Networks are composed of Long Short-Term Memory layers leading to a single activation node. They predict that a word is only complex if the activation node outputs a value equal or bigger than 0.5. The architecture of their networks is determined through cross-validation over the joint dataset. While one of their systems consist on the best performing Neural Network architecture found (Baseline), the other combines the five best architectures using an eXtreme gradient boosted ensemble (Combined) (Nat, 2016).

TALN Uses Random Forests to perform CWI. While one of their systems is trained over the joint dataset (RandomForest_SIM), the other is trained over the decomposed dataset (RandomForest_WEI), and includes the number of annotators that deemed the word to be complex as a feature. Both systems also include various lexical, morphological, semantic and syntactic features (Ronzano et al., 2016).

USAAR Presents two Bayesian Ridge classifiers. Their first system (Entropy) is trained based solely on a hand-crafted Word Sense Entropy metric, which is calculated for each target word in the joint dataset. Their other system (Entroplexity) combines Word Sense Entropy with perplexity measures calculated with a language model (Martínez Martínez and Tan, 2016).

UWB Performs CWI with the help of Maximum Entropy classifiers. Both classifiers use only one feature: document frequencies of words in Wikipedia. While one of them is trained over the joint dataset (All), the other is trained over the decomposed dataset (Agg) (Konkol, 2016).

Results

The final G and F-score ranks obtained by each system are reported in the first two columns of Table 3.8. The systems that have achieved the highest G-scores are our SV000gg systems, which combine various Threshold-Based, Lexicon-Based and Machine Learning approaches

with minimalistic voting techniques. Similarly, the system from the TALN team, which has the third highest G-score, also uses an Ensemble method of Random Forests.

One of the most clearly highlighted phenomena in our results is the recurring effectiveness of Decision Trees and Random Forests in CWI: out of the systems with the 10 best G-scores, only three do not employ them. Their reliability is also highlighted by the variety of distinct feature sets used to train them, which range from morphological to syntactic features. In contrast, the scores obtained by the BHASHA-DECISIONTREE and GARUDA-HSVM&DT systems reveal that Decision Trees and Random Forests can be much less effective when incorporated in more elaborate setups.

When it comes to F-score, Decision Trees and Random Forests remain dominant among the top 10 systems, but ultimately take the back seat to a much more minimalistic Threshold-Based strategy. The PLUJAGH-SEWDFF system, which has obtained the highest F-score, simply learns the threshold of word frequencies in Wikipedia that maximises the F-score over the joint dataset. Similarly, the LTG systems, which achieved the second and third highest F-scores, use Decision Trees to learn a threshold over the number of annotators that judged a word to be complex.

Another interesting finding refers to the difference between raw word frequencies and single-word language model probabilities. The systems submitted by the PLUJAGH team, which learn thresholds over raw word frequencies from Simple Wikipedia, have consistently outperformed the "(TB) Simple Wiki" baseline, which uses language model probabilities, in both G and F-scores.

Perhaps the biggest surprise from our results comes from the overall poor performance of systems which employ Neural Networks and/or word embedding models: all systems that do so, which are the ones submitted by team AI-KU, AmritaCEN, CoastalCPH and Sensible, have among them ranked no better than 18th in G-score and 30th in F-score. This comes as a disappointing result, given that these techniques and resources have been successfully employed in state-of-the-art approaches to a wide variety of tasks in recent years. We hypothesise that the small amount of training data available is the main cause for their unsatisfactory performance.

3.2 Substitution Selection

In the usual LS pipeline, after a word is classified as complex, a Substitution Generation (SG) system produces candidate substitutions for it. However, in most LS approaches (Glavaš and Štajner, 2015; Horn et al., 2014; Kajiwara et al., 2013), these candidates are produced in context-independent fashion i.e. the SG system is not aware of the content surrounding the

G	F	Team	System	Acc.	Prec.	Rec.	F-score	G-score
1	13	SV000gg	Soft	0.779	0.147	0.769	0.246	0.774
2	16	SV000gg	Hard	0.761	0.138	0.787	0.235	0.773
3	9	TALN	RandomForest_WEI	0.812	0.164	0.736	0.268	0.772
4	10	UWB	All	0.803	0.157	0.734	0.258	0.767
4	11	PLUJAGH	SEWDF	0.795	0.152	0.741	0.252	0.767
4	15	JUNLP	NaiveBayes	0.767	0.139	0.767	0.236	0.767
5	7	HMC	RegressionTree05	0.838	0.182	0.705	0.290	0.766
6	5	HMC	DecisionTree25	0.846	0.189	0.698	0.298	0.765
7	12	JUNLP	RandomForest	0.795	0.151	0.730	0.250	0.761
8	8	MACSAAR	RFC	0.825	0.168	0.694	0.270	0.754
9	6	TALN	RandomForest_SIM	0.847	0.186	0.673	0.292	0.750
10	14	MACSAAR	NNC	0.804	0.146	0.660	0.240	0.725
11	21	Pomona	NormalBag	0.604	0.095	0.872	0.171	0.714
12	22	UWB	Agg	0.569	0.089	0.885	0.161	0.693
13	23	Pomona	GoogleBag	0.568	0.088	0.881	0.160	0.691
14	24	IIIT	NCC	0.546	0.084	0.880	0.154	0.674
15	2	LTG	System2	0.889	0.220	0.541	0.312	0.672
16	18	MAZA	A	0.773	0.115	0.578	0.192	0.661
18	30	Sensible	Baseline	0.591	0.078	0.713	0.140	0.646
19	29	ClacEDLK	ClacEDLK-RF_0.6	0.688	0.081	0.548	0.141	0.610
20	1	PLUJAGH	SEWDFF	0.922	0.289	0.453	0.353	0.608
21	31	IIIT	NCC2	0.465	0.071	0.860	0.131	0.604
22	25	ClacEDLK	ClacEDLK-RF_0.5	0.751	0.090	0.475	0.152	0.582
24	4	MAZA	B	0.912	0.243	0.420	0.308	0.575
25	34	AmritaCEN	w2vecSim	0.627	0.061	0.486	0.109	0.547
26	23	GARUDA	SVMPP	0.796	0.099	0.415	0.160	0.546
27	38	AIKU	native1	0.583	0.057	0.512	0.103	0.545
27	39	AIKU	native	0.555	0.056	0.535	0.101	0.545
28	40	AKTSKI	wsys	0.587	0.056	0.490	0.100	0.534
28	41	AKTSKI	svmbasic	0.512	0.053	0.558	0.097	0.534
29	19	BHASHA	DECISIONTREE	0.836	0.118	0.387	0.181	0.529
30	17	USAAR	entropy	0.869	0.148	0.376	0.212	0.525
31	33	Sensible	Combined	0.737	0.072	0.390	0.122	0.510
32	20	BHASHA	SVM	0.844	0.119	0.363	0.179	0.508
33	35	CoastalCPH	NeuralNet	0.693	0.063	0.398	0.108	0.506
34	3	LTG	System1	0.933	0.300	0.321	0.310	0.478
35	28	USAAR	entroplexity	0.834	0.097	0.305	0.147	0.447
36	40	AmritaCEN	w2vecSimPos	0.743	0.060	0.306	0.100	0.434
38	26	GARUDA	HSVM&DT	0.880	0.112	0.226	0.149	0.360
39	34	CoastalCPH	Concatenation	0.869	0.080	0.171	0.109	0.285

Table 3.8 Results of the Complex Word Identification task of SemEval 2016

complex word being simplified. The step responsible for deciding which of the generated candidates can replace the complex word in the context in which it appeared is Substitution Selection (SS).

As discussed in (Paetzold and Specia, 2013), the perfect selector would be able to reliably answer the following two questions about each candidate substitution:

1. Can it replace the target word without compromising the sentence's grammaticality?

2. Can it replace the target word without changing the sentence's meaning?

Given that both these questions have two possible answers (Yes or No), it seems natural that one way to approach SS would be by training binary classifiers over annotated data containing examples of good and bad replacements. We could not, however, find any examples of SS strategies in literature that attempt to do so.

In an effort to gather new insight on what separates good from bad word replacements, we have conducted a user study with 400 fluent speakers of English. In this study, volunteers were asked to judge the grammaticality and meaning preservation aspect of several candidate substitutions for various complex words. Using the annotated data produced in the study, we introduce new datasets for SS, which allow the training of supervised selectors. The following Sections describe the user study in more detail.

Notice that we skip the step of Substitution Generation in our user studies. We do so because we believe that the experiments of Horn et al. (2014) and De Belder and Moens (2012a) already provide with sufficient insight with respect to the relationship between Substitution Generation and the needs of non-native English speakers.

3.2.1 Data Sources

We first created a list of $1,471$ complex words by filtering any numbers, names, colours and stop words from the ones obtained in the CWI study. We then produced an average of 50 candidate substitutions for each word by combining the output of all SG systems in the LEXenstein framework (Paetzold and Specia, 2015), which exploit complex-to-simple parallel corpora (Horn et al., 2014), word embedding models (Glavaš and Štajner, 2015; Paetzold and Specia, 2016i), WordNet (Biran et al., 2011; Devlin and Tait, 1998) and the Merriam Dictionary[4]. For more details on how each of these approaches work, please refer to the LEXenstein documentation presented in Chapter 8.3.

Using the strategy described by Paetzold and Specia (2015), we selected the 10 candidates with the highest cosine similarity to each complex word, given a typical bag-of-words

[4]http://www.merriam-webster.com

(CBOW) word embeddings model with 500 dimensions trained over a corpus of 7 billion words extracted from various sources (Mikolov et al., 2013a). Using the Text Adorning module of LEXenstein, we ensure that all candidates have the same conjugation form as the complex word itself.

Finally, we extract, according to availability, up to three sentences from Wikipedia in which each of these complex words appear (2,554 in total), and create 25,540 annotation instances by replacing the complex word in each sentence with one of the 10 candidate substitutions selected.

3.2.2 Annotation Process

400 fluent English speakers participated, all university students and staff. We did not require volunteers to be non-native speakers of English, since the complex words selected were deemed so by themselves and, in order for someone to judge the quality of an alternative for a complex word, it needs to understand its meaning. Each was presented with 80 annotation instances accompanied by the original complex word for reference. For each instance, they were presented with three hypothesis:

- The substitution preserves the sentence's GRAMMATICALITY.
- The substitution preserves the sentence's MEANING.
- None of the above.

Volunteers were obliged to select at least one option, but could not select neither of the first two along with "None of the above". The resulting dataset contains 25,540 annotated instances in total. For agreement analysis purposes, 1,600 instances were annotated by 5 volunteers each, while the remaining 23,940 instances were annotated by a single volunteer.

3.2.3 Nature of Good Substitutions

We have calculated several features to compare the grammatical and/or meaning preserving substitutions against the remaining substitutions. Table 3.9 illustrate the average and standard deviation feature values of candidates annotated positively by at least three out of five annotators (Good), and not (Bad), with respect to their grammaticality, meaning preservation, and both of them jointly. We chose six features:

- Language model probabilities of the sentence with the candidate in place of the target, given four 3-gram language models trained over the SubIMDB (Paetzold and Specia,

2016i), SUBTLEX (Brysbaert and New, 2009b) and Simple Wikipedia (Kauchak, 2013) corpora. Language model sentence probabilities have been used in the creation of some of the most effective Lexical Simplification systems in literature (Glavaš and Štajner, 2015; Horn et al., 2014; Paetzold and Specia, 2016i). Language models were trained with SRILM.

- The cosine word vector similarity between the candidate and the target (Target Sim.), as well as the average cosine similarity between the vector of the candidate and the vectors of content words in the sentence (Context Sim.). These features have been used in the creation of the unsupervised lexical simplifier of Glavaš and Štajner (2015). The embeddings model was trained with word2vec (Mikolov et al., 2013a) with the CBOW architecture and 500 dimensions over a corpus of 7 billion words extracted from various sources (Brysbaert and New, 2009b; Kauchak, 2013; Paetzold, 2015b).

- The probability of the candidate of receiving the same POS tag attributed to the target (POS Prob.). This feature has shown to be a strong indicator of grammaticality (Aluisio and Gasperin, 2010; Nunes et al., 2013). The POS tag conditional probability models were trained over dependency parses produced by the Stanford Parser (Klein and Manning, 2003) over the NewsCrawl corpus[5].

The column following the average values for Good and Bad candidates contain • for features for which it was found a statistically significant difference between the averages through an F-test ($p < 0.01$), and ∘ for the remainder. The results suggest that, even though they are able to account for context, n-gram language model probabilities are much less effective in distinguishing good from bad candidates than the word vector distance between target and candidate. Nonetheless, the same cannot be observed for the average similarity between candidate and context words.

Another interesting finding from our results that agree with previous contributions (Aluisio and Gasperin, 2010; Nunes et al., 2013) regards the probability of the candidate receiving the POS tag of the target (POS Prob.), which does indeed show a strong relationship with grammaticality.

We have also found that, out of the 356 candidates judged both grammatical and meaning preserving by at least three annotators, 171 (48%) are not listed in WordNet as either synonyms, hypernyms or hyponyms of the target word. This suggests that simplification strategies such as the ones of Devlin and Tait (1998) and Biran et al. (2011), which extract candidate substitutions of complex words from WordNet, can suffer from low coverage.

[5]http://www.statmt.org/wmt11/translation-task.html

	Grammaticality			Meaning			Joint (G/M)		
Feature	Good	Bad	p	Good	Bad	p	Good	Bad	p
P. Subimdb	-0.9 ± 0	-1.0 ± 0	○	-1.0 ± 0	-0.9 ± 0	○	-0.9 ± 0	-1.0 ± 0	•
P. Subtlex	-3.1 ± 1	-3.2 ± 1	•	-3.2 ± 1	-3.2 ± 2	•	-3.1 ± 1	-3.4 ± 2	○
P. Simple	-4.2 ± 1	-4.3 ± 2	•	-4.2 ± 2	-4.3 ± 2	•	-4.2 ± 2	-4.4 ± 2	○
Trgt Sim.	0.4 ± 0.2	0.2 ± 0.2	•	0.3 ± 0.2	0.2 ± 0.2	•	0.3 ± 0.2	0.2 ± 0.2	•
Ctxt Sim.	0.1 ± 0.1	0.0 ± 0.1	○	0.1 ± 0.1	0.0 ± 0.1	○	0.1 ± 0.1	0.0 ± 0.1	•
POS Prob.	0.6 ± 0.4	0.4 ± 0.4	•	0.5 ± 0.4	0.4 ± 0.4	○	0.5 ± 0.4	0.3 ± 0.4	•

Table 3.9 Average and standard deviation of features of those words which were deemed grammatical, meaningful, or both by at least 3 annotators, and those that were not.

3.2.4 Annotation Agreement

The average Kappa inter-annotator agreement scores for the data in this user study are 0.391 ± 0.16 for grammaticality, 0.424 ± 0.16 for meaning preservation, and 0.450 ± 0.16 for both of them jointly. Inspecting the data, we found that most disagreements resulted from situations in which the target word was part of a multi-word expression. Take, for example, the target word *turn* in the sentence "*That in turn makes it difficult to affect policies to curb distracted driving*", which, in this case, is part of the multi-word expression *in turn*. Annotators were very much divided on whether or not candidate *reverse* preserved either grammaticality or meaning in this case: some judged it to be neither grammatical nor meaningful, while others claimed the very opposite.

3.2.5 Evaluating Binary Classification Methods

In this experiment, we use the data produced in our user study to assess the effectiveness of several binary classification strategies in discerning between good and bad candidate substitutions.

Approaches

The approaches compared in this experiment can be divided in three categories:

- **Threshold-Based (TB)**: Given a certain feature, we find the threshold t that best separates good from bad replacements. We consider all features described in Section 3.2.3. To find t, an exhaustive search was performed on the training set over $10,000$ equally distant values in the interval between the minimum and maximum value of each feature.
- **Machine-Learning Assisted (MA)**: Using various Machine Learning techniques, we train classifiers to decide whether or not a given candidate substitution is appropriate. We consider five techniques:

1. **Multi-Layer Perceptron**: Trained over a categorical cross-entropy loss and Stochastic Gradient Descent, with three layers with 64 dimensions.
2. **Linear Perceptron**: A typical linear model trained with the perceptron algorithm.
3. **Support Vector Machines**: Trained with Radial Basis Function kernel and an L2 regulariser.
4. **Decision Trees**: Trained with a maximum entropy splitting function.
5. **Random Decision Forests**: Composed of 50 Decision Trees trained with a maximum entropy splitting function.

The features used are the same used by the Threshold-Based approaches. We use Keras to train our Multi-Layer Perceptron model, and scikit-learn to train all others. The hyper-parameters of all classifiers, including the number and size of hidden layers in the Multi-Layer Perceptron, kernels, regularisers and splitting functions, were optimised through 10-fold cross validation.

- **Baselines (BA)**: We include the "All Good" and "All Bad" baselines, which predict that all candidate substitutions are good and bad, respectively.

Datasets

From the data collected in our user study, we extracted annotations pertaining to each of the following three word replacement properties:

- **Grammaticality**: Considers only the judgments made with respect to the grammaticality of a candidate substitution. A candidate receives label 1 if a total of at least n people has judged it to preserve the grammaticality of a sentence, and 0 otherwise.
- **Meaning Preservation**: Considers only the judgments made with respect to the meaning preservation of a candidate substitution. A candidate receives label 1 if a total of at least n people has judged it to preserve the meaning of a sentence, and 0 otherwise.
- **Appropriateness**: Considers the judgments with respect to grammaticality and meaning preservation jointly. A candidate receives label 1 if a total of at least n people has judged it to be both grammatical and meaning preserving, and 0 otherwise.

For each property, we then create training sets using two settings:

- **Optimistic**: A candidate receives label 1 if at least one annotator has judged a given property to be true, and 0 otherwise.
- **Conservative**: A candidate receives label 1 if at least three annotators have judged a given property to be true, and 0 otherwise.

The training sets are comprised of all the $1,600$ instances annotated by five volunteers. In order to create test sets, we use the remaining instances, which have been annotated by only one volunteer. We create one test set for each of the aforementioned word replacement properties: a candidate receives label 1 if a given property was judged to be true by the annotator, and 0 otherwise.

Evaluation

For evaluation, we use the traditional Accuracy (A), Precision (P), Recall (R), and F-score (F).

Results

The results obtained for all properties and settings are presented in Tables 3.10 through 3.15.

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.567	0.568	0.993	0.722
TB	Prob. SUBTLEX	0.567	0.568	0.992	0.722
TB	Prob. Simple	0.566	0.567	0.994	0.722
TB	POS Tag Probability	0.643	0.658	0.771	0.710
TB	Target Similarity	0.568	0.568	1.000	0.724
TB	Context Similarity	0.568	0.568	1.000	0.724
MA	Multi-Layer Perceptron	0.568	0.568	1.000	0.724
MA	Decision Trees	0.600	0.589	0.977	**0.735**
MA	Random Trees	0.583	0.578	0.983	0.728
MA	Support Vector Machines	0.584	0.587	0.902	0.711
MA	Linear Perceptron	0.553	0.570	0.872	0.689
BA	All Bad	0.432	0.000	0.000	0.000
BA	All Good	0.568	0.568	1.000	0.724

Table 3.10 Results for grammaticality in the *optimistic* setting

It can be noticed that, overall, Threshold-Based approaches offer very competitive performance to that of Machine Learning techniques. Nonetheless, their performance is, in most scenarios, very similar to that of the "All Good" baseline.

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.567	0.568	0.993	0.722
TB	Prob. SUBTLEX	0.567	0.568	0.992	0.722
TB	Prob. Simple	0.566	0.567	0.994	0.722
TB	POS Tag Probability	0.646	0.666	0.755	0.708
TB	Target Similarity	0.568	0.568	1.000	0.724
TB	Context Similarity	0.568	0.568	0.999	0.724
MA	Multi-Layer Perceptron	0.568	0.568	1.000	0.724
MA	Decision Trees	0.637	0.633	0.857	**0.728**
MA	Random Trees	0.632	0.642	0.794	0.710
MA	Support Vector Machines	0.610	0.625	0.779	0.694
MA	Linear Perceptron	0.530	0.570	0.698	0.628
BA	All Bad	0.432	0.000	0.000	0.000
BA	All Good	0.568	0.568	1.000	0.724

Table 3.11 Results for grammaticality in the *conservative* setting

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.341	0.339	0.993	0.505
TB	Prob. SUBTLEX	0.342	0.339	0.992	0.505
TB	Prob. Simple	0.340	0.338	0.993	0.505
TB	POS Tag Probability	0.461	0.349	0.686	0.463
TB	Target Similarity	0.344	0.340	0.998	**0.507**
TB	Context Similarity	0.339	0.339	1.000	0.506
MA	Multi-Layer Perceptron	0.661	0.000	0.000	0.000
MA	Decision Trees	0.342	0.339	0.993	0.505
MA	Random Trees	0.407	0.345	0.835	0.488
MA	Support Vector Machines	0.339	0.339	1.000	0.506
MA	Linear Perceptron	0.404	0.327	0.721	0.450
BA	All Bad	0.661	0.000	0.000	0.000
BA	All Good	0.339	0.339	1.000	0.506

Table 3.12 Results for meaning preservation in the *optimistic* setting

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.429	0.340	0.725	0.463
TB	Prob. SUBTLEX	0.468	0.340	0.609	0.437
TB	Prob. Simple	0.452	0.337	0.641	0.442
TB	POS Tag Probability	0.470	0.352	0.672	0.462
TB	Target Similarity	0.545	0.406	0.740	**0.524**
TB	Context Similarity	0.358	0.343	0.980	0.508
MA	Multi-Layer Perceptron	0.661	0.000	0.000	0.000
MA	Decision Trees	0.570	0.358	0.341	0.349
MA	Random Trees	0.632	0.393	0.157	0.225
MA	Support Vector Machines	0.339	0.339	1.000	0.506
MA	Linear Perceptron	0.612	0.350	0.170	0.229
BA	All Bad	0.661	0.000	0.000	0.000
BA	All Good	0.339	0.339	1.000	0.506

Table 3.13 Results for meaning preservation in the *conservative* setting

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.243	0.240	0.992	0.386
TB	Prob. SUBTLEX	0.244	0.240	0.991	0.387
TB	Prob. Simple	0.242	0.240	0.992	0.386
TB	POS Tag Probability	0.464	0.278	0.769	0.408
TB	Target Similarity	0.247	0.242	0.998	0.389
TB	Context Similarity	0.241	0.240	1.000	0.388
MA	Multi-Layer Perceptron	0.240	0.240	1.000	0.387
MA	Decision Trees	0.459	0.277	0.777	**0.409**
MA	Random Trees	0.438	0.267	0.768	0.397
MA	Support Vector Machines	0.481	0.270	0.679	0.386
MA	Linear Perceptron	0.309	0.237	0.844	0.370
BA	All Bad	0.760	0.000	0.000	0.000
BA	All Good	0.240	0.240	1.000	0.387

Table 3.14 Results for appropriateness in the *optimistic* setting

Cat.	Approach	A	P	R	F
TB	Prob. SubIMDB	0.384	0.241	0.724	0.361
TB	Prob. SUBTLEX	0.450	0.244	0.617	0.350
TB	Prob. Simple	0.429	0.241	0.638	0.349
TB	POS Tag Probability	0.496	0.286	0.730	0.411
TB	Target Similarity	0.514	0.300	0.770	**0.432**
TB	Context Similarity	0.241	0.240	1.000	0.388
MA	Multi-Layer Perceptron	0.760	0.000	0.000	0.000
MA	Decision Trees	0.646	0.278	0.297	0.287
MA	Random Trees	0.733	0.318	0.099	0.151
MA	Support Vector Machines	0.240	0.240	1.000	0.387
MA	Linear Perceptron	0.416	0.248	0.705	0.367
BA	All Bad	0.760	0.000	0.000	0.000
BA	All Good	0.240	0.240	1.000	0.387

Table 3.15 Results for appropriateness in the *conservative* setting

Among Machine Learning approaches, Decision Trees are clearly the most consistent throughout the different scenarios. But like our Threshold-Based approaches, the performance of all Machine Learning techniques is very close to that of the "All Good" baseline, which suggests that, in most scenarios, the strategies evaluated do not effectively learn how to distinguish between good and bad candidates, and rather just acquire a strong bias towards declaring all candidates good.

In order to discover how these binary classifiers perform as part of an LS system in practice, we have conducted a benchmarking with them. The description and results of this experiment can be found in Appendix A.

3.3 Substitution Ranking

After candidate substitutions have been generated and selected, they can be given to Substitution Ranking (SR), a step of which the goal is ensuring that the replacement selected for a complex word is as simple as possible. As discussed in Chapter 2, SR approaches vary from minimalistic frequency-based approaches (Carroll et al., 1998; Devlin and Tait, 1998) to more sophisticated supervised strategies (Horn et al., 2014), and most of which do not target a specific audience.

In this user study, in which 300 non-native English speakers took part, we investigate how simplicity is perceived by them. Volunteers were presented with several sentences with a "gap" and two words to fill it, and were asked to select which one of them made the sentence easier to understand. The data produced led to a new dataset, and the findings of our

experiments contradict previous claims on word simplicity. The following Sections describe the user study in more detail.

3.3.1 Data Sources

We have extracted all 901 sentences from the ones in our Substitution Selection user study which had a minimum of 2 and maximum of 4 candidates annotated as both grammatical and meaning preserving by at least three annotators. In order for its simplicity to also be assessed, we added the target complex word of each sentence to the set of candidates. We then replaced the target word in each sentence with a gap marker, and created an annotation instance for each pair of candidates, totalling $4,200$ pairs (438 sentences * 3 pairs (3 candidates including target) + 436 * 6 pairs (4 candidates including target) + 27 * 10 pairs (5 candidates including target)).

3.3.2 Annotation Process

300 non-native English speakers participated as annotators, all university students and staff. Volunteers provided anonymous information about their native language, age, education level and English proficiency level. Each volunteer was presented with 70 annotation instances, each composed of a sentence with a gap, and two candidates to fill it with.

For each instance, volunteers were asked to select which candidate made the sentence easier to understand. Volunteers could also select a third option, in case both candidates made the sentence equivalently complex/simple. All $4,200$ instances were annotated by 5 volunteers.

Once instances were annotated, we used the algorithm introduced by Wauthier et al. (2013) to infer rankings from binary comparisons for all 901 sets of candidate substitutions. Their algorithm takes as input a matrix M of dimensions NxN, where N is the number of candidates in a given ranking problem, and each cell $M[i, j]$ is the probability of candidate c_i having a better rank i.e. being simpler than c_j. We calculate each cell of matrix M using Equation 3.3, in which C is the count of occurrences of a given type of judgment made by the volunteers.

$$M[i, j] = \frac{C\big(c_i \text{ is simpler than } c_j\big)}{C\big(c_i \text{ is simpler than } c_j\big) + C\big(c_j \text{ is simpler than } c_i\big)} \tag{3.3}$$

Once matrix M is produced, the score of each candidate is inferred using Equation 3.4.

$$R(c_i) = \sum_{i \neq j} (2M[i,j] - 1) \tag{3.4}$$

The details of the mathematical model that leads to Equation 3.4 can be found in the work of Wauthier et al. (2013). Finally, candidates are ranked in decreasing order of score.

3.3.3 Profile of Annotators

The annotators of this user study are speakers of 39 languages. The most predominant languages were Portuguese (18.5%), Chinese (11.1%) and Spanish (10%). Annotators are between 18 and 63 years old (average 26.7). 55% of the volunteers were Postgraduate students, 33.8% Undergraduate, and 11.2% were in High School. 40.8% claimed to have Advanced (C2) English proficiency skills, 31.6% Pre-Advanced (C1), 23.3% Upper-Intermediate (B2), 3.5% Intermediate (B1), and 0.6% Pre-Intermediate (A2).

3.3.4 Dataset Analysis

To understand which characteristics of candidates make them simpler, we computed the correlation between the simplicity rankings and 24 features:

- Word Length.
- Number of syllables. Syllables were extracted with the Morph Adorner toolkit (Burns, 2013).
- Number of senses in WordNet.
- 3-gram language model probability from the Brown (Francis and Kucera, 1979), SUBTLEX (Brysbaert and New, 2009b), SubIMDB (Paetzold and Specia, 2016i), Wikipedia and Simple Wikipedia (Kauchak, 2013) corpora. Language models were trained with the help of SRILM (Stolcke, 2002).
- Probabilities from a 3-gram language model trained over Simple Wikipedia of eight n-grams with (l,r) tokens to the left (l) and right (r) of the candidate.
- The conditional probability of the candidate given the POS tag of the target word. The models and parameters used are the same described in Section 3.2.3.
- The cosine distance between the embedding vectors of the target word and the candidate. The models and parameters used are the same described in Section 3.2.3.

- The average cosine distance between the vector of the candidate and the content words in the sentence. The models and parameters used are the same described in Section 3.2.3.

The values in Table 3.16 reveal that, while word length and number of syllables correlate poorly with word complexity, simpler words tend to be more ambiguous and occur more frequently in corpora. These findings reinforce the ones from our CWI user study.

More importantly, our results show that correlation scores of (1,1) (one token to the left and right) n-grams consistently outperform the scores of single-word frequencies (0,0) in all metrics used. These findings contradict a long-standing assumption that context is not an important factor in word simplicity estimation (Biran et al., 2011; Carroll et al., 1999; Devlin and Tait, 1998; Rello et al., 2013b; Shardlow, 2013a).

We also found that the target complex word itself was the simplest candidate in 46.4% of the instances, highlighting the need for more effective Complex Word Identification and Substitution Generation methods for simplification.

Feature	Spearman	Pearson
Word Length	-0.176 ± 0.6	-0.153 ± 0.6
Number of Syllables	-0.099 ± 0.6	-0.088 ± 0.5
Frequency: SubIMDB	0.408 ± 0.6	0.358 ± 0.5
Frequency: SUBTLEX	0.450 ± 0.5	0.394 ± 0.5
Frequency: Simple Wiki	0.479 ± 0.5	0.428 ± 0.5
Frequency: Wikipedia	0.458 ± 0.5	0.405 ± 0.5
Frequency: Brown	0.363 ± 0.6	0.316 ± 0.5
Sense Count	0.328 ± 0.6	0.292 ± 0.5
Synonym Count	0.272 ± 0.6	0.241 ± 0.6
Hypernym Count	0.275 ± 0.6	0.244 ± 0.5
Hyponym Count	0.293 ± 0.6	0.257 ± 0.5
Minimum Sense Depth	-0.207 ± 0.6	-0.185 ± 0.5
Maximum Sense Depth	0.126 ± 0.6	0.109 ± 0.6
N-gram $(1,0)$	$\mathbf{0.506 \pm 0.5}$	$\mathbf{0.453 \pm 0.5}$
N-gram $(0,1)$	0.478 ± 0.5	0.421 ± 0.5
N-gram $(1,1)$	0.489 ± 0.5	0.430 ± 0.5
N-gram $(2,0)$	0.505 ± 0.5	0.451 ± 0.5
N-gram $(2,1)$	0.492 ± 0.5	0.432 ± 0.5
N-gram $(0,2)$	0.477 ± 0.5	0.420 ± 0.5
N-gram $(1,2)$	0.487 ± 0.5	0.428 ± 0.5
N-gram $(2,2)$	0.487 ± 0.5	0.428 ± 0.5
POS Prob.	0.130 ± 0.6	0.108 ± 0.6
Target Sim.	0.247 ± 0.6	0.219 ± 0.6
Context Sim.	0.260 ± 0.6	0.227 ± 0.5

Table 3.16 Simplicity feature correlation scores

3.3.5 Annotation Agreement

The average Kappa inter-annotator agreement scores for this user study also resemble the ones reported in Section 3.1.3: although the agreement between all annotators is quite encouraging (0.454 ± 0.05), the scores are even higher for annotators with similar backgrounds. Annotators within the same education level, age band and proficiency levels reach agreement scores of 0.468 ± 0.01, 0.482 ± 0.02 and 0.486 ± 0.01, respectively. But much like what was observed in our user study on Complex Word Identification, the highest agreement comes from annotators that speak the same native language (0.601 ± 0.15). This serves as further evidence that one's native language plays an important role in vocabulary acquisition.

3.3.6 Performance Comparison

In this experiment, we assess the performance of various Substitution Ranking strategies over the ranks produced in our user study.

Approaches

We compare the performance of three rankers from literature:

- **Biran** (Biran et al., 2011): Employs the metric in Equation 8.4, in which $F(c,C)$ is the frequency of candidate c in corpus C, and $\|c\|$ its length.

$$M(c) = \frac{F(c, \text{Wikipedia})}{F(c, \text{Simple Wikipedia})} \times \|c\| \tag{3.5}$$

- **Horn** (Horn et al., 2014): Uses Support Vector Machines Joachims (2002) to learn a ranking model from data with several word and n-gram frequency features.

- **Glavas** (Glavaš and Štajner, 2015): Ranks candidates according to several features, such as n-gram frequencies and word vector similarity with the target word, and then re-ranks them according to their average rankings.

For the Biran, Horn and Glavas rankers, we use the same parameters described in the publications in which they were introduced. We optimise the parameters of all supervised rankers through 10-fold cross validation. For completeness, we also include a metric-based ranker for each one of the 24 features listed in Section 3.3.4. In total, we compare 27 ranking strategies.

Datasets

We first compile the 901 ranking instances produced in our user study into a dataset, which is then split into a training and a test set, containing 450 and 451 instances, respectively. The training set was used in the training of supervised rankers, and the test set was used for evaluation.

Metrics

For evaluation, we use the TRank-at-n metric proposed by Specia et al. (2012), which measures the proportion of times in which a candidate with a gold-standard rank equal to $r \leq n$ was ranked first.

Results

The results in Table 3.17 reveal that the N-gram $(2,0)$ ranker offers the highest TRank-at-1, 2 and 3 scores, outperforming all other systems. Surprisingly, the Horn ranker, which is the only one to learn a ranking model from data, was outperformed by much simpler unsupervised strategies.

The unsatisfactory performance of the Horn ranker highlights the need for more suitable supervised ranking models. Although unsupervised rankers can be more easily adapted to different languages, supervised rankers can learn how to automatically combine different features in order for the rankings produced to best suit the needs of a target audience.

3.4 Conclusions

In this Chapter, we have described three user studies with the goal of providing new resources and insights on the simplification needs of non-native English speakers.

In our Complex Word Identification study we learned that words which do not challenge non-native speakers are much more present in corpora, while those which do challenge them tend to be much less ambiguous. In contrast with what was reported by Rello et al. (2013b) in experiments with dyslexic readers, we found no evidence of a relationship between the non-native English speakers' perception of complexity and either word length or number of syllables. Our experiments also suggest that while their English proficiency level indicates how many words will challenge them, their native language indicates which words these will be. Using the data produced in this user study, we launched the Complex Word Identification task of SemEval 2016. From the task, we have learned that our Performance-Oriented Soft Voting strategy is the most effective approach among all 42 submitted.

Ranker	TRank-at-1	TRank-at-2	TRank-at-3
Word Length	0.373	0.678	0.849
Number of Syllables	0.339	0.676	0.816
Frequency: SubIMDB	0.519	0.798	0.904
Frequency: SUBTLEX	0.075	0.366	0.485
Frequency: SimpleWiki	0.082	0.341	0.490
Frequency: Wikipedia	0.534	0.812	0.916
Frequency: Brown	0.503	0.812	0.916
Sense Count	0.494	0.792	0.904
Synonym Count	0.448	0.758	0.870
Hypernym Count	0.455	0.747	0.870
Hyponym Count	0.419	0.723	0.874
Minimum Sense Depth	0.392	0.716	0.833
Maximum Sense Depth	0.235	0.574	0.720
N-gram $(1,0)$	0.581	0.847	**0.958**
N-gram $(0,1)$	0.545	0.834	0.933
N-gram $(1,1)$	0.572	0.836	0.937
N-gram $(2,0)$	**0.592**	**0.851**	**0.958**
N-gram $(2,1)$	0.574	0.836	0.937
N-gram $(0,2)$	0.552	0.827	0.929
N-gram $(1,2)$	0.570	0.834	0.941
N-gram $(2,2)$	0.579	0.838	0.941
POS Prob.	0.333	0.639	0.774
Target Sim.	0.477	0.761	0.879
Context Sim.	0.466	0.758	0.879
Biran	0.404	0.725	0.866
Horn	0.565	0.831	0.933
Glavas	0.588	0.831	0.937

Table 3.17 Performance comparison of rankers over the data produced in this user study

The user study on Substitution Selection has allowed us to better outline the characteristics of good substitutes of complex words. It was found that they tend to take the same grammatical role as the complex word, and also that they tend to appear closely in the distributional space of word embedding models. From our agreement analysis, we learned that single-word replacements tend to compromise the perception of multi-word expressions.

To complete our investigation, we presented a user study on the task of Substitution Ranking. We found further evidence that, unlike ambiguity indicators and language model probabilities, length and number of syllables have little to do with word simplicity for non-native speakers of English. N-gram probabilities proved the most reliable simplicity indicators among the features evaluated, which contradicts the assumption often made in earlier contributions that context offers no important clues on a word's simplicity. Through a benchmark of Substitution Ranking strategies, we have also highlighted the need for more effective supervised rankers that can learn the needs of a target audience.

Table 3.18 summarises the data produced from each of these studies. Some examples of models and applications that could be built from these datasets are readability assessment tools, semantic analysers, text profilers and full lexical simplifiers.

User Study	Sentences	Words	Word Pairs	Annotators	Annotations
Complex Word Identification	9,200	87,244	-	400	158,624
Substitution Selection	2,554	25,540	-	400	31,940
Substitution Ranking	901	3,193	4,200	300	21,000
Total	**12,655**	**115,977**	**4,200**	**1,100**	**211,564**

Table 3.18 Summary of annotated data produced: number of unique sentences, words and word pairs annotated, as well as the number of annotators who participated and annotations produced.

Chapter 4

Joint Lexical Simplification

Pipelined Lexical Simplification approaches have changed a lot throughout the years. The early approach of Devlin and Tait (1998) used very minimalistic techniques and resources: an early version of WordNet to extract synonyms of complex words and the Brown corpus for Kucera-Francis coefficients. As discussed in the survey of Chapter 2, these have evolved into much more sophisticated strategies, which use complex-to-simple parallel corpora (Horn et al., 2014), supervised ranking strategies (Paetzold, 2015b) and word embedding models (Glavaš and Štajner, 2015). These modern strategies have led to substantial increases in performance (Horn et al., 2014), and have been used in almost all research in the field.

However, work is still needed to find effective approaches that model all steps of the LS pipeline jointly. Although there are some reasonably successful examples of joint Text Simplification models that treat it as a Machine Translation problem (Specia, 2010; Wubben et al., 2012), there are no joint models that focus strictly on LS.

Joint models have been successful in various Natural Language Processing tasks, specially with the popularization of Recurrent Neural Networks (RNNs) (Goller and Kuchler, 1996). In Text Generation, for example, early systems were composed by very elaborate pipelines with steps such as "Planning" and "Realisation" (Reiter et al., 2000), which had to be modelled individually. These have recently given space to much simpler strategies. Sutskever et al. (2011) introduces Recurrent Neural Networks that require nothing but large corpora of text to learn how to produce texts of a given topic. By simply providing the network with a portions of text t and requiring for it to predict subsequent portions $t+1$, it learns how to create original content of a certain type.

An even more notable example of task in which the shift towards joint models can be noticed is Machine Translation. Statistical translation approaches, such as Hierarchical Phrase-Based (Chiang, 2005) and Syntax-Based (Koehn et al., 2003) models, usually work in a two-step process: calculating translation probabilities from large parallel corpora (or

"alignment"), and searching for the best translation for a given sentence (or "decoding"). Although their effectiveness have attracted the attention of big companies such as Google (Allauzen et al., 2014), these models are not very proficient in capturing long-distance dependencies, mostly due to the fact that the steps of alignment and decoding are performed independently Marcu and Wong (2002). The strategy described in (Bahdanau et al., 2014), which is often referred to as "Neural Machine Translation", offers an alternative: it uses Recurrent Neural Networks to jointly model the entire translation process (alignment, decoding, etc.). The architecture they use is composed of an encoder and a decoder. While the encoder is responsible for encoding the training sentences in the source language, the decoder is responsible for decoding it into the target language. Their results show that this simplistic take to translation can outperform even consolidated statistical models.

In this Chapter, we describe our efforts in creating joint LS models with Recurrent Neural Networks. To accomplish so, we make use of Neural Language Models, which not only jointly model Substitution Generation, Selection and Ranking, but also allow for its parameters to be optimized over a validation corpus that represents the needs of non-native English speakers.

4.1 Neural Language Models for Lexical Simplification

Word frequencies and statistical language models are the most frequently used resources in LS. While single word frequencies have been shown to correlate well with the non-native English speakers' perception of simplicity (Specia et al., 2012), n-gram probabilities have been incorporated in various context-aware ranking strategies that aim to capture grammaticality and meaning preservation (Glavaš and Štajner, 2015; Horn et al., 2014; Paetzold, 2015b).

Language models suit the task of LS so well that, from a theoretical standpoint, they could be a self-contained approach to joint Lexical Simplification. In simple terms, given a sequence of words $W = w_1, ..., w_m$ of any size, a language model calculates a probability $P(W)$ that determines the relative likelihood of W pertaining to a certain language. To learn a probability distribution, n-gram models, for example, rely on two assumptions:

1. **The Markov assumption:** The probability of a word in a sequence is conditioned only on preceding words i.e. the sequence of words to its left.

2. **The independence assumption:** The probability of a word is conditioned only on the $n-1$ words to its left (hence the name "n-gram" model).

An n-gram model will extract counts of n-grams with size $\leq n$ from a given corpus and calculate $P(w_1,...,w_m)$ using Equation 4.1, in which n is the order of the model (4-gram, 5-gram, etc) and *count* the number of times an n-gram has appeared in the training corpus.

$$P(w_1,...,w_m) = \prod_{i=1}^{m} P\left(w_i \mid w_{i-(n-1)},...,w_{i-1}\right) = \prod_{i=1}^{m} \frac{count\left(w_{i-(n-1)},...,w_{i-1},w_i\right)}{count\left(w_{i-(n-1)},...,w_{i-1}\right)} \quad (4.1)$$

Notice that probability $P(w_1,...,w_m)$ is decomposed into n-gram probabilities based on the two aforementioned assumptions. This probability decomposition is used to make calculations tractable and reduce sparsity within the model. In order to calculate the probability of partially unseen sequence of words, smoothing techniques, such as the one of Chen and Goodman (1999), are used. These attempt to "predict" the n-gram probability of unseen sequences.

Assuming that word and n-gram frequencies are sufficient indicators of grammaticality, meaning preservation and word simplicity, if a certain n-gram language model somehow perfectly captures the needs of a target audience, it could be used as an LS approach that models the Substitution Generation, Selection and Ranking steps jointly. Given a target complex word in a sentence S, one could replace it with every word in a vocabulary, then simply use probability $P(S)$ to determine which one of the vocabulary words is the most likely to fit the context of the complex word. Notice that this approach would still require a Complex Word Identification strategy to determine which words should be simplified.

Although elegant in its formulation, this strategy is not practical. Because n-gram models are not parametrised, it becomes difficult to incorporate information about the needs of a target audience during training. If an n-gram language model were to be trained over corpora created specifically to assist readers who are unfamiliar with a language, such as Simple Wikipedia (Kauchak, 2013), there would still be no guarantee that it would capture the intricate needs of non-native English speakers. The experiments of Horn et al. (2014) and Paetzold and Specia (2015) suggest so: although n-gram language model probabilities from Simple Wikipedia can help in the creation of some of the most effective LS strategies to date, they are outperformed by probabilities from larger corpora that do not contain simplified content exclusively, such as the Google 1T corpus Evert (2010).

Recurrent Neural Networks have been successfully applied to language modelling in an effort to address the limitations of n-gram models caused by the Markov and independence assumptions, such as their ineffectiveness in capturing long-distance relationships between words. As discussed in Sutskever et al. (2014), RNNs can receive as input sequences, such as sentences or image pixels, learn from the entirety of the information contained in them, and

either output a single label, in the case of "sequence-to-label" networks, or an entire sequence of labels, in the case of "sequence-to-sequence" networks. The fact that RNNs can receive an entire sequence as input has made it an extremely appealing for language modelling.

One of the first "Neural Language Models" (NLM), which is still very popular, was introduced by Mikolov et al. (2010). It creates a language model by training an RNN that learns how to predict the next word to appear after an input sequence of words. In simpler terms, their RNN learns to answer the question "Which word comes next in this incomplete sentence?", which makes it a sequence-to-label network. The architecture used in their approach is illustrated in Figure 4.1, which was extracted directly from Mikolov et al. (2010).

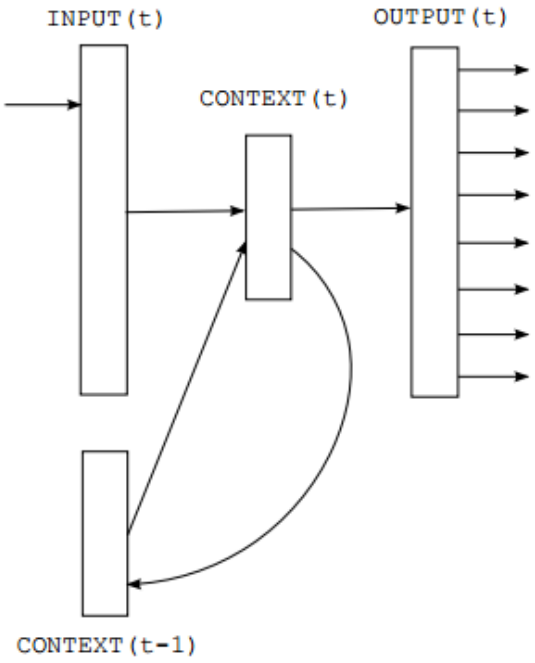

Fig. 4.1 Neural Network architecture for a language model

In Figure 4.1, t represents a time-step, and $t-1$ the time-step that came before it. In their architecture, the input layer receives a one-hot representation of the word present at a certain position in the input sequence. In this context, a one-hot representation is a sparse vector with the same size as the vocabulary of the text over which the network is to be trained. All its values are 0 except in the position that represents a given word, which has value 1. The context layer, on the other hand, represents the output produced by a series of hidden layers for a certain time-step. Finally, the output layer provides the probability of each word in the vocabulary being the next one to follow. In order to calculate probability P using this model, one can simply retrieve the highest probability observed in the output layer. Notice that, in

order to propagate information through time, the context layer is concatenated to the input layer after each time-step. In other words, the prediction made in time-step t uses as input the output of the hidden layer in time-step $t-1$. The advantages of using this approach as opposed to traditional n-gram models for LS come primarily because of two factors:

1. It does not rely on either the Markov or the independence assumptions.
2. Its training is an entirely parametrised learning procedure.

Factor number one allows the model to disregard the need to calculate the likelihood of an arbitrarily sized sequence of words as the product of the likelihood of smaller preceding n-grams, which consequently allows it to capture long-distance dependencies more effectively (Arisoy et al., 2012). This is very important for LS because often times it is only possible to determine which word best fits a context after assessing the entirety of the sentence in question. Factor number two even more important. In a parametrised learning process, one can maximise parameter values based on a validation corpus. This means that, unlike n-gram models, Neural Language Models allow for parameters to be selected so that they maximise the probabilities of sentences that represent the needs of a target audience.

With our language modelling strategy selected, we can formalise a joint Lexical Simplification strategy that targets the needs of non-native English speakers. To joint model LS, we first train a Neural Language Model through the following steps:

1. Acquire a large training corpus of text.
2. Acquire a validation corpus containing sentences that are generally understandable by non-native English speakers.
3. Train an RNN over the training corpus, maximising parameter values over the validation dataset until convergence.

Once the RNN is trained, one can then move on to simplification. Given a target complex word w_t in a sentence S and a vocabulary V, perform the following steps:

1. For each candidate in $c \in V$, create a new version of S by replacing w_t with c.
2. Calculate the probability $P(S)$ for all $c \in V$.
3. Rank the candidates according to P.
4. Replace w_t with the c that has produced the highest probability P.

Notice that, using this strategy, we discard the need for any explicit generation or selection steps, and instead model the entire LS task as a ranking problem. In the Sections that follow, we describe the various experiments conducted with our novel approach to LS.

4.2 Experimental Settings

In our experiments, we compare the performance of our approach under various distinct settings with other consolidated LS approaches in literature.

4.2.1 Validation Corpora

We train three Neural Language Models for our strategy, each with one of the following validation corpora:

- **LexMTurk**: The LS evaluation dataset introduced by Horn et al. (2014). Each instance in LexMTurk is composed of a sentence, a target complex word, and simpler candidate substitutions suggested by English speakers from the U.S. In order to create a validation corpus from LexMTurk, we transform its instances into a series of sentences. For each instance, we create one sentence for each distinct candidate substitution suggested by the annotators by replacing the target complex word with it. The LexMTurk validation corpus is composed of $6,697$ sentences.

- **SSEval**: The Substitution Selection datasets produced in the user study of Section 3.2. The SSEval dataset is composed of the concatenation of all training and testing instances in which the candidate word being judged was deemed both grammatical and meaning preserving by at least one of the annotators. Given that the SSEval instances are composed of a sentence, an original target complex word and candidate substitution, we create validation sentences by replacing the target word with the substitution in each instance. The resulting corpus is composed of $9,303$ sentences.

- **All**: Composed by the concatenation of both aforementioned validation corpora. It contains $16,000$ sentences.

By using these different validation corpora, we hope to be able to outline if maximising parameters over SSEval, which was created based on the needs of non-native speakers of English, offers any advantages over using LexMTurk, which is a more general LS dataset. All four NLMs use the same training corpus, which is composed of texts extracted from the UMBC webbase[1], News Crawl[2], and the SUBTLEX (Brysbaert and New, 2009b) corpora, as well as Wikipedia and Simple Wikipedia (Kauchak, 2013). The vocabulary used is composed of all words which appear at least 10 times in the training corpus.

[1] http://ebiquity.umbc.edu/resource/html/id/351
[2] http://www.statmt.org/wmt11/translation-task.html

4.2.2 Architecture

We determine the architecture of the NLMs through 5-fold cross validation over each validation corpus. The metric used is the average perplexity (Brown et al., 1992b) of all sentences in the validation corpus. The perplexity of a sentence $w_1, ..., w_m$ is calculated as described in Equation 4.2, where $P(w_1, ..., w_m)$ is the probability of sentence $w_1, ..., w_m$ estimated by a given NLM.

$$\text{perplexity}\,(w_1, ..., w_m) = \sqrt[m]{\frac{1}{P(w_1, ..., w_m)}} \tag{4.2}$$

The NLM architecture parameters optimised are:

- **Number of layers:** From 1 to 5.
- **Size of layers:** From 64 to 256 in intervals of 64.
- **Learning rate:** From 0.1 to 0.00001 in intervals of one order of magnitude.

4.2.3 Pipelined Approaches

We compare our approach to three other simplifiers from literature:

- Devlin (Devlin and Tait, 1998): The first lexical simplifier Devlin and Tait (1998). Its approaches to each step of the pipeline are:
  - **Substitution Generation**: Extracts synonyms from WordNet.
  - **Substitution Selection**: Does not perform Substitution Selection.
  - **Substitution Ranking**: Uses frequencies from the Brown corpus Francis and Kucera (1979).
- Kauchak (Horn et al., 2014): One of the most effective supervised lexical simplifiers in literature. Its approaches to each step of the pipeline are:
  - **Substitution Generation**: Extracts complex-to-simple word correspondences from word alignments between Wikipedia and Simple Wikipedia.
  - **Substitution Selection**: Does not perform Substitution Selection.
  - **Substitution Ranking**: Learns a ranking model using Support Vector Machines Joachims (2002) from the examples in the LexMTurk dataset.

For training, we use the same parameters and resources used in the experiments of Section 3.2.5.

- Glavas (Glavaš and Štajner, 2015): An entirely unsupervised system that performs similarly to the Horn simplifier. Its approaches to each step of the pipeline are:

  - **Substitution Generation**: Extracts the 10 words closest to a given target complex word in a word embeddings model.
  - **Substitution Selection**: Does not perform Substitution Selection.
  - **Substitution Ranking**: Ranks candidates using the average ranking obtained for various semantic and collocational metrics.

  For training, we use the same parameters and resources used in the experiment of Section 3.2.5. Notice that, like our NLM approach, none of the aforementioned strategies perform a Complex Word Identification step.

4.2.4 Datasets

To evaluate our simplifier, we use the NNSeval dataset, described in Section 8.4. This is the evaluation dataset created for the benchmarking experiment described in Chapter 8.5, and accounts for the needs of non-native English speakers. It contains 239 instances, each composed by a sentence, a word deemed complex by a non-native English speaker, and a set of gold-standard candidate replacements.

4.2.5 Evaluation Metrics

The metrics used are the ones introduced by Horn et al. (2014), which are:

- **Precision**: The proportion of instances in which the target word was replaced with any of the gold-standard candidates in the dataset, including the target word itself.
- **Accuracy**: The proportion of instances in which the target word was replaced with any of the gold-standard candidates in the dataset, except for the target word itself.
- **Changed Proportion**: The proportion of times in which the target word was replaced with any different word.

4.3 Joint vs. Pipelined

In our first experiment, we compare how our joint NLMs fair against consolidated pipelined approaches. The performance scores obtained by them are illustrated in Table 4.1.

Simplifier	Precision	Accuracy	Changed Proportion
Devlin	0.335	0.117	0.782
Kauchak	0.364	**0.172**	0.808
Glavas	**0.590**	0.163	0.573
NLM(LexMTurk)	0.016	0.013	0.996
NLM(SSEval)	0.017	0.014	0.996
NLM(All)	0.017	0.013	0.996

Table 4.1 Joint simplification versus pipelined LS performance comparison

The results suggest that joint models with NLMs are not a suitable approach to LS: under all configurations, our NLMs were able to correctly simplify only 1.3% of the complex words in NNSeval. Through F-tests, we found no statistically significant difference ($p > 0.05$) between the output produced by the different configurations of NLMs, which reveals that it cannot effectively capture the needs of a target audience during training.

Inspecting the output, we were able to get a better insight on what the biggest limitations of our approach are. The main reason for its poor performance seems to be the large amount of spurious candidates that produce high probabilities P. Consider, for example, the instance in NNSeval in which "*tender*" must be simplified in the sentence "*The grilled octopus was very tender.*". The highest ranking candidates produced by NLM (All) for this problem are "*good*", "*important*", "*nice*", "*dangerous*" and "*smart*". One can notice that, although all of these candidates fit the sentence grammaticaly, they do not capture the meaning of "*tender*" accurately enough. We hypothesise this is caused by two main factors:

1. In order to avoid modelling Substitution Generation explicitly, our approach considers all words in a very large vocabulary as a possible candidate, and hence allows for too many spurious words to be ranked higher than other more useful alternatives.

2. By simply replacing the target complex word with each candidate, our approach entirely discards any type of information pertaining to the semantic relationship between them, hence making it impossible, in some cases, to correctly determine which candidates would capture the meaning convened by the target.

Interestingly, both of these limitations can be addressed if we disregard our intent to joint model the entire LS process and instead adapt our approach to fit the traditional pipeline. They

could, for example, be solved if we were to pair our NLMs with a Substitution Generation approach. The generator would significantly reduce the amount of spurious candidates to be considered, which would consequently allow for our models to focus strictly in the task of Substitution Ranking. The inverse could also help. Our models could serve as a Substitution Generation approach by selecting, for example, the 10 highest ranked candidates, which could then be re-ranked by a context-aware Substitution Ranking approach that takes into consideration the relationship between the candidates and the target word. The following experiments test these hypotheses.

4.4 Neural Networks for Substitution Ranking

In our second experiment, we test the hypothesis that if we take our NLMs as an approach to Substitution Ranking and pair it with a Substitution Generation approach, we improve its performance. To do so, instead of considering all words in the vocabulary to be a possible candidate, we use the ones produced by each of the generators used by the baseline simplifiers.

The results illustrated in Table 4.2 show the simplification performance scores from our previous experiment, along with the scores for 12 other systems. Each of these systems is composed by the generation approach used by either the Devlin, Kauchak or Glavas simplifier, and one of our NLMs.

Simplifier	Precision	Accuracy	Changed Proportion
Devlin	0.335	0.117	0.782
Kauchak	0.364	**0.172**	0.808
Glavas	**0.590**	0.163	0.573
NLM(LexMTurk)	0.017	0.013	0.996
NLM(SSEval)	0.017	0.013	0.996
NLM(All)	0.017	0.013	0.996
Devlin+NLM(LexMTurk)	0.314	0.130	0.816
Devlin+NLM(SSEval)	0.314	0.130	0.816
Devlin+NLM(All)	0.314	0.130	0.816
Kauchak+NLM(LexMTurk)	0.372	0.146	0.774
Kauchak+NLM(SSEval)	0.360	0.146	0.774
Kauchak+NLM(All)	0.372	0.146	0.774
Glavas+NLM(LexMTurk)	0.301	0.126	0.824
Glavas+NLM(SSEval)	0.293	0.130	0.824
Glavas+NLM(All)	0.301	0.126	0.824

Table 4.2 Performance of Neural Language Models as Substitution Ranking approaches

Even though F-tests did not find any statistically significant difference ($p > 0.05$) between the output produced by different validation corpora, the results agree with our hypothesis: combining NLMs with Substitution Generation strategies makes them a much more reliable approach to LS. Nonetheless, our models were only able to outperform the much simpler frequency-based ranker used by the Devlin simplifier in Accuracy, and have lead to noticeable losses in performance in other settings.

4.5 Neural Networks for Substitution Generation

In our final experiment, we test the hypothesis that if we take our NLMs as an approach to Substitution Generation and pair it with a different Substitution Ranking approach, we improve its performance. To do so, instead of using our models to select only the word in the vocabulary that yields the highest probability P, we use them to select the 10 best candidates, then forward them to the rankers used by our baseline simplifiers for re-ranking.

Simplifier	Precision	Accuracy	Changed Proportion
Devlin	0.335	0.117	0.782
Kauchak	0.364	**0.172**	0.808
Glavas	**0.590**	0.163	0.573
NLM(LexMTurk)	0.017	0.013	0.996
NLM(NNSeval)	0.017	0.013	0.996
NLM(SSEval)	0.017	0.013	0.996
NLM(All)	0.017	0.013	0.996
NLM(LexMTurk)+Devlin	0.000	0.000	1.000
NLM(NNSeval)+Devlin	0.000	0.000	1.000
NLM(SSEval)+Devlin	0.000	0.000	1.000
NLM(All)+Devlin	0.000	0.000	1.000
NLM(LexMTurk)+Kauchak	0.046	0.013	0.967
NLM(NNSeval)+Kauchak	0.050	0.013	0.962
NLM(SSEval)+Kauchak	0.046	0.013	0.967
NLM(All)+Kauchak	0.046	0.013	0.967
NLM(LexMTurk)+Glavas	0.172	0.008	0.837
NLM(NNSeval)+Glavas	0.180	0.008	0.828
NLM(SSEval)+Glavas	0.172	0.008	0.837
NLM(All)+Glavas	0.172	0.008	0.837

Table 4.3 Performance of Neural Language Models as Substitution Generation approaches

The results of Table 4.3 show the simplification performance scores from our first experiment, along with the scores of 12 system combinations. Each of these systems is

composed by our NLM generation approach and the Substitution Ranking strategy used by either the Devlin, Kauchak or Glavas simplifier.

The results do not allow us to either conclusively confirm or refuse our hypothesis. By pairing our generators with the Kauchak and Glavas rankers, we have managed to obtain a small, yet statistically significant ($p < 0.05$), improvement in Precision, but not Accuracy. In contrast, the frequency-based Devlin ranker has led to 0% Precision and Accuracy, given that it does not account for any type of context-related information during ranking. Overall, replacing the baseline simplifiers' generators with our NLM alternatives has led to considerable losses in performance with respect to all metrics used. As in our previous experiment, F-tests did not find any statistically significant difference ($p > 0.05$) between the validation corpora used.

4.6 Conclusions

In this Chapter, we introduced a Lexical Simplification approach that jointly models the Substitution Generation, Selection and Ranking steps in the typical LS pipeline. Our joint LS approach exploits Neural Language Models, which predict the probability of a sequence of words without the need for decomposing it into n-grams.

In order to discard the need of modelling Substitution Generation and Selection explicitly, we simply consider all words in the vocabulary to be a possible candidate, then rank them according to the probability they yield when replacing a complex word in the sentence it was found. This strategy allows us to address Lexical Simplification as a ranking problem. By using Neural Language Models instead of n-gram models, we are also able to incorporate validation data that represents the needs of a target audience during training.

Through experimentation, however, we found that NLMs do not work well for this problem in practice. Our approach managed to correctly simplify complex words only 1.3% of the time, which is over 13 times less than the best performing pipelined simplifier. It was only by combining our approach with the traditional LS pipeline that we have managed to improve its performance. By pairing NLMs with the Substitution Generation approaches used by the baseline simplifiers, we managed to reduce the amount of spurious candidates considered during ranking, and hence create a better simplifier. We found no statistically significant difference between NLM variations validated over different corpora, which suggests that, in the context of LS, these models cannot be effectively customised to the needs of a target audience during training.

Overall, we found no evidence that joint modelling various steps in the LS pipeline with NLMs is a sensible alternative to addressing each step of the pipeline individually. Unlike

what was achieved already in many other Natural Language Processing tasks, we have yet to find an effective way of employing RNNs in LS, specially when it comes to joint modelling multiple steps in the pipeline. In what follows, we present our efforts in creating an effective pipelined approach to Lexical Simplification.

Chapter 5

Substitution Generation with Word Embeddings

Substitution Generation (SG) is the task of, given a set of n target complex words $\{t_1, ..., t_n\}$, generating a set of candidate substitutions C_i for each target word t_i, such that every candidate c_j in C_i is a word, multi-word expression or phrase that can replace t_i in one of the possible contexts in which it may appear. In a typical pipelined LS approach, a given candidate c_i does not need to be able to replace t_i in every possible context, nor it needs to be simpler than t_i, given that such judgments are to be performed by the steps of Substitution Selection and Ranking. SG can also be performed in context-aware fashion. The generator can produce different candidates for "*roll*" as a verb and as a noun, for example.

The main goal of SG is to facilitate the process of Substitution Selection by discarding words and expressions that could not possibly replace a given complex word in any context. In Table 5.1 we present some examples of valid candidate substitutions extracted from WordNet's synonyms for various words.

Target Word	Generated Candidates
yield	concede, grant, proceeds, production, payoff, succumb
pitch	flip, toss, sky, tar, delivery, slake, rant, lurch
roll	revolve, hustle, pluck, bun, peal, scroll, cast, coil
treat	dainty, delicacy, goody, handle, process, cover, plow

Table 5.1 Examples of generated candidate substitutions from WordNet

One can notice that all target words in Table 5.1 are highly ambiguous and can assume various grammatical forms, such as noun or verb, depending on the context. Words such as these often have several distinct senses and synonyms, and would consequently require a Substitution Selection approach to be very effective to prevent an LS system from per-

forming erroneous substitutions. Such observation highlights the importance of Substitution Generation in LS: even though fetching WordNet synonyms is a simple strategy, it would still require much less effort for a selector to choose the appropriate substitutions from the candidates in Table 5.1 than from the entirety of the English vocabulary.

As discussed in Chapter 2, most SG approaches extract candidate substitutions from linguistic databases, such as WordNet, BabelNet[1] and UMLS (Bodenreider, 2004), or from corpora composed by documents in their original forms aligned to simpler versions, such as Wikipedia and Simple Wikipedia (Kauchak, 2013). Such approaches rely, however, on annotated data which is either scarce or non-existent for various languages. Although in recent years there have been several efforts to create linguistic databases for languages other than English, they are not numerous and still very expensive to produce.

In order to address the limitations of SG strategies in literature, we propose a novel approach to the task, which generates candidate substitutions without using either linguistic databases or parallel data. In our experiments, we compare the performance of our approach to that of numerous others in not only SG alone (Section 5.6), but also in practice by pairing it with various Substitution Selection and Ranking strategies in order to create a complete LS system (Section 5.7).

5.1 Substitution Generation with Word Embeddings

In Natural Language Processing, one of the main reasons why Neural Networks have become so frequently used is the popularisation of word embedding models, in which words are represented as numerical vectors (or "word embeddings"). Such models can be trained in various ways, and usually require only large corpora of text for training. Modern word embedding models are also very flexible, and allow for the dimensionality of the numerical vectors that represents words to be selected during training (Mikolov et al., 2013a).

Strategies that exploit word embedding models have been shown very effective in tasks such as Machine Translation (Zou et al., 2013), Sentiment Analysis (Glorot et al., 2011), Question Answering (Iyyer et al., 2014) and many others. Although the first mentions of such models date back to 1986 (Rumelhart et al., 1988), researchers have only been able to successfully exploit their use in the aforementioned tasks by employing the models recently proposed by Mikolov et al. (2013a). They are:

- **Continuous Bag-of-Words (CBOW)**: Attempts to estimate the best numerical vector for each and every word in a vocabulary V, based on the words that surround them in

[1] http://babelnet.org

the various contexts in the training corpus. The CBOW model is trained with the use of a Neural Network composed of three layers: the input, the hidden, and the output layer. Figure 5.1, extracted from (Rong, 2014), shows the architecture of a CBOW Neural Network in its simplest form, in which numerical vectors are estimated without taking into account the different contexts in which a given word is inserted. The components in Figure 5.1 can be described as such:

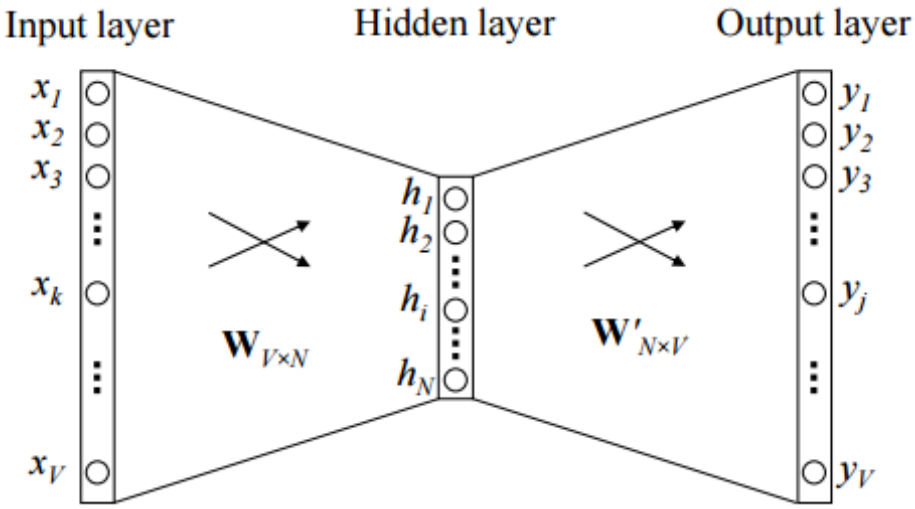

Fig. 5.1 Neural Network used in the training of a CBOW model

- V: The size of the vocabulary present in the training corpus.
- N: The size of the embedding vectors.
- $\mathbf{x}$: A one-hot vector representation of size V of a given input word.
- $\mathbf{W}$: The matrix between the input and hidden layers that transforms the one-hot $\mathbf{x}$ vector of an input word into its numerical vector representation $\mathbf{h}$ of size N.
- $\mathbf{h}$: The numerical vector produced from the product between $\mathbf{x}$ and $\mathbf{W}$.
- $\mathbf{W}'$: The matrix between the hidden and output layers that transforms the numerical vector representation h of an input word back into an output vector of size V.
- $\mathbf{y}$: The output vector produced from the product between $\mathbf{h}$ and $\mathbf{W}'$.

In order for the Neural Network to be able to produce meaningful numerical vectors $\mathbf{h}$, the values in $\mathbf{W}$ and $\mathbf{W}'$ are updated iteratively with respect to a loss function. During each iteration, the values in $\mathbf{W}$ are used to produce numerical vectors, while the values

of $\mathbf{W}'$ are used in the scoring function of Equation 5.1, which determines how the weights of $\mathbf{W}$ and $\mathbf{W}'$ are going to be updated.

$$u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h} \tag{5.1}$$

In Equation 5.1, u_j is the score of word w_j, and $\mathbf{v}'_{w_j}$ is the jth column of matrix $\mathbf{W}'$. Using this scoring function, one can produce the loss function described in Equation 5.2, in which j^* is the gold-standard output word expected.

$$E = -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \tag{5.2}$$

By deriving the value of E with respect to u_j, one is able to derive the update functions for $\mathbf{W}'$ and $\mathbf{W}$, and hence obtain a Neural Network that produces meaningful numerical vector representations for words. One can notice, however, that this architecture does not take into account the context of the words in the training corpus. To accommodate the words' context, Mikolov et al. (2013a) proposed the enhanced architecture illustrated in Figure 5.2, extracted from (Rong, 2014).

In this enhanced architecture, the input layer contains multiple nodes that share the same matrix $\mathbf{W}$. To estimate the numerical vector $\mathbf{h}$, the Neural Net averages the vectors resulting from the product between the various one-hot vectors $\mathbf{x}_i$ and matrix $\mathbf{W}$.

- **Continuous Skip-Gram (Skip-Gram)**: Like the CBOW model, it also attempts to estimate the best numerical vector for each and every word in a vocabulary V, based on the various contexts in which they appear. The Skip-Gram model, however, uses a "reversed" Neural Network architecture to estimate the numerical vectors: a word's context is incorporated in the output, as opposed to the input layer. During training, while the Neural Network of the CBOW model is updated with respect to how proficient it is in predicting a word based on its context, the Neural Network used by the Skip-Gram model is updated with respect to its proficiency in predicting a word's context based on the word itself. Figure 5.3, extracted from (Rong, 2014), shows the Neural Network architecture used in the training of a Skip-Gram model.

  In Figure 5.3, each vector $\mathbf{y}_i$ is a vector of dimension V that corresponds to a given context word. Notice that all $\mathbf{y}$ vectors share the same matrix $\mathbf{W}'$. In this setup, the scoring function u is calculated through the same equation used for the CBOW model, but the loss function, which described in Equation 5.3, is calculated differently. In

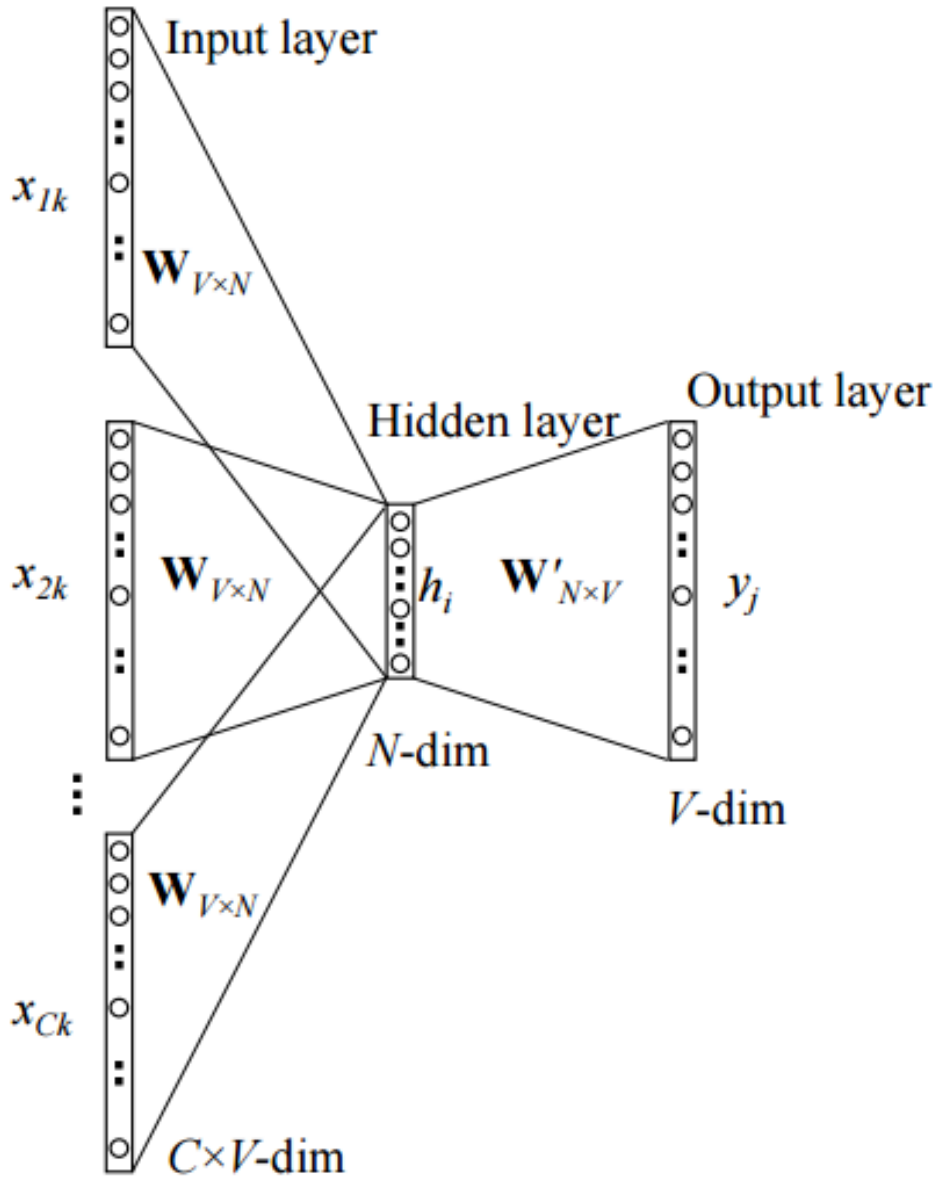

Fig. 5.2 Neural Network used in the training of a context-aware CBOW model

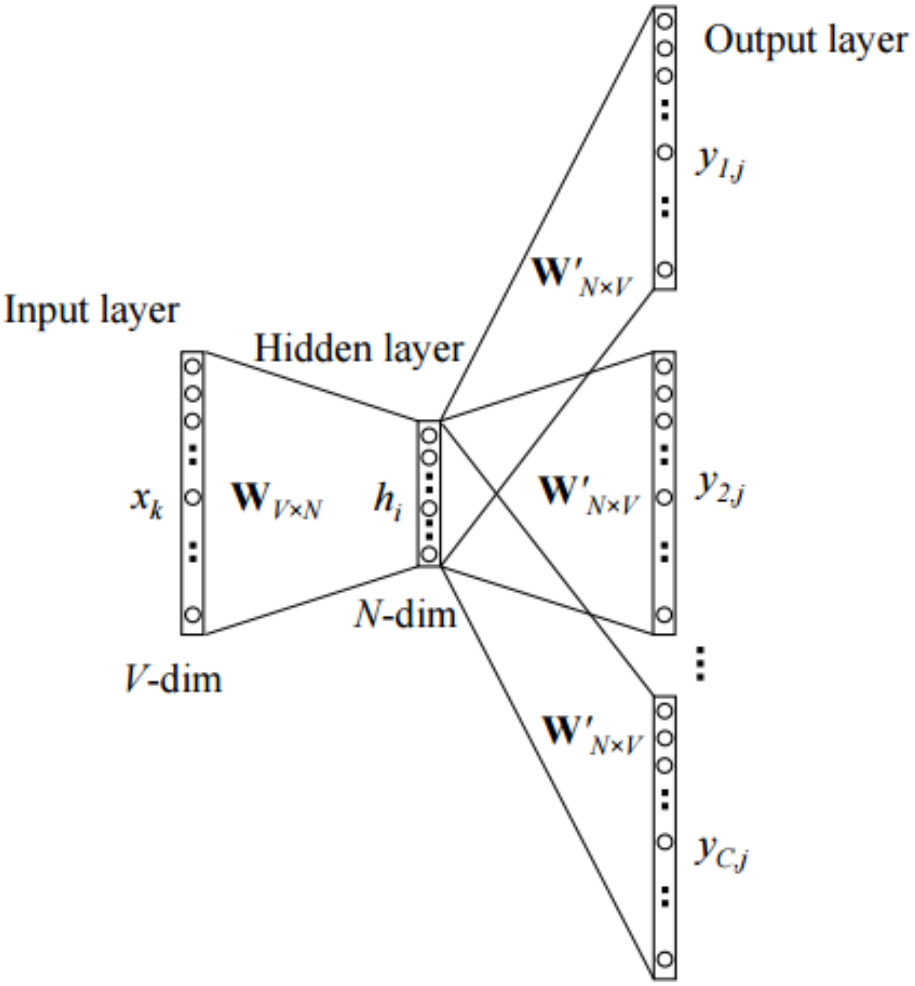

Fig. 5.3 Neural Network used in the training of the Skip-Gram model

Equation 5.3, C is the number of context words to be predicted at the output layer, and j_c^* is the index of the actual output word expected for the cth position of the input's context.

$$E = -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \tag{5.3}$$

Perhaps the most remarkable property of word embeddings are the complex relations between words that they are able to capture. Mikolov et al. (2013b) shows that word embeddings are an effective approach to problems that require analogical reasoning: if one were to take the numerical vector that represents the word "*Berlin*", deducted the value of "*Germany*" and added the value of "*France*", one would find that the word whose vector is closest to the resulting numerical vector is "*Paris*". Another example (introduced by Mikolov et al. (2013c)) of analogical reasoning that can be inferred from word embeddings refers to gender: if one were to deduct the numerical vector that represents "*man*" from "*king*", and then add to it the vector of "*woman*", the resulting vector would be closest to that of "*queen*". Such examples serve as evidence that word embeddings are a reliable source for the extraction of words with related meaning, and hence show that such models can be an interesting resource to be used in Substitution Generation.

In order to explore the use of word embeddings in SG, we propose a new algorithm for the extraction of candidate substitutions. Our approach is very simple, and requires only a stemmer and a word embeddings model trained over large corpora. Given a set of target words $\{t_1, t_2 ... t_{n-1}, t_n\}$, where each t_i pertains to a sentence $S_i \in \{S_1, S_2 ... S_{n-1}, S_n\}$, two selection limits l and m, and a model M, such that $M(t_i)$ returns the embedding vector of t_i, our algorithm extracts m substitution candidates for each target word t_i by performing the following steps:

1. For each target word $t_i \in \{t_1, t_2 ... t_{n-1}, t_n\}$, extract the l candidates c_j in model M which have the shortest cosine distances between $M(t_i)$ and $M\left(c_j\right)$.

2. Discard any candidate c_j that has the same stem as t_i. Discard any candidate c_j of which the stem is a sub-string of t_i. Discard any candidate c_j which is a sub-string of t_i. Discard any candidate c_j which is a super-string of t_i. Discard any candidate c_j which is a super-string of the stem of t_i.

3. For the remaining candidates of each target word $t_i \in \{t_1, t_2 ... t_{n-1}, t_n\}$, extract the m candidates c_j which have the shortest cosine distances between $M(t_i)$ and $M\left(c_j\right)$.

Notice that, while l limits the amount of words in the vocabulary that will be considered for filtering, m determines how many candidates will be retrieved for each target word. The first step of this algorithm avoids the need to stem all words in the vocabulary during the second step, which could be very time consuming.

The algorithm above explores the hypothesis that words which have similar numerical vectors have a similar meaning, and hence may be synonyms, hypernyms, hyponyms, antonyms or related in other ways (e.g. "*cat*" and "*dog*", "*black*" and "*white*", etc). The step 2 applies filters that attempt to avoid for redundant substitutions to be generated.

Typical word embedding models have, nevertheless, a limitation that may compromise their application in SG: they do not account for the ambiguity of words. In such models, every possible meaning of a word is represented by a single numerical vector, which in turn makes it impossible for our algorithm to predict neither which of a target word's possible interpretations the generated candidates will refer to. To address this issue, we propose a new strategy to enhance word embedding models for Lexical Simplification. We describe our strategy in detail in the following Section.

5.2 Context-Aware Word Embedding Models

In order for a word embeddings model M to offer support for word ambiguity, it would have to represent every single possible sense s_i of a target word t with an unique numerical vector $M(t, s_i)$. Perhaps the most minimalistic way to learn such a model would be, instead of re-thinking the entirety of the mathematical formulation behind word embedding models, to simply disambiguate each and every word in every sentence of the training corpus, and then annotate them with their respective meanings. Table 5.2 illustrates some examples of sentences of which the ambiguous words are annotated with synset identifiers from WordNet.

Its birthstone is_Synset('be.v.12') the diamond_Synset('ball_field.n.01').
It has_Synset('have.v.11') 31 days_Synset('day.n.02').
August is_Synset('be.v.12') named_Synset('diagnose.v.01') for Augustus Caesar.
They can rollback_Synset('rollback.n.02') changes_Synset('change.n.08').
What did_Synset('do.v.08') the artist_Synset('artist.n.01') value_Synset('value.n.02')?

Table 5.2 Sentences annotated with synset identifiers

By training a model over the corpus produced by this process, one would learn an embeddings model that distinguishes between all senses of a given word. Such strategy, however, works under two very strong assumptions:

1. The number of senses of all words in a vocabulary can be reliably quantified.
2. The word sense disambiguator can proficiently distinguish between the senses of all words in every sentence of the training corpus.

However, recent contributions in literature provide evidence that such assumptions are not true. The statistics made available in WordNet's website[2] state that WordNet, which is one of the largest ontologies currently available for the English language, have sense annotations for only $155,287$ out of the $291,500$ words present in the Oxford Dictionary[3]. The results reported in the work of Basile et al. (2014), which introduces a state-of-the-art approach for Word Sense Disambiguation, show that even the most effective disambiguators make mistakes almost 30% of the time.

Given the limitations of both disambiguators and the data over which they are trained, we suggest a more robust strategy for the challenge of creating context-aware embedding models. Our strategy addresses the aforementioned limitations by:

- **Using Part-of-Speech tags instead of senses**: Although they are not as informative as senses in word disambiguation, POS tags offer some insight on the meaning of a word: highly ambiguous words can often assume various grammatical roles, such as that of a noun or a verb, depending on the context in which they are inserted. POS tags are also much less sparse than word senses: while WordNet has a total of $117,659$ registered word senses, the Penn Treebank tagset is composed of only 36 tags (Marcus et al., 1993).
- **Replacing disambiguators with Part-of-Speech taggers**: While state-of-the-art disambiguators can achieve no more than 72% of accuracy in most datasets, the performance of modern POS taggers is much more impressive, reaching upwards of 97% accuracy in consolidated datasets (Manning, 2011).

By employing these two compromises, we impart more feasibility to our strategy, but annotating the words in the training corpus with raw POS tags in Treebank format would also impart some unnecessary sparsity in our model. Consider, for example, the words "*pitching*" and "*pitched*" present in the sentences "*The player is pitching fast balls at over* 40*mph*"

[2] http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html
[3] http://public.oed.com/history-of-the-oed/dictionary-facts

and "*The fastest fast ball ever pitched in history reached* 50*mph*". Although "*pitching*" and "*pitched*" are both verbs and share the same meaning, they would likely assume two distinct POS tags if the aforementioned sentences were to be tagged: that of a verb in the gerund (VBG), and the past tense (VBD). The same phenomena may also occur between nouns in singular (NN) and plural (NNS) forms, adjectives and adverbs in their original (JJ/RB), superlative (JJR/RBR) and comparative (JJS/RBS) forms, and many other grammatical classes.

To prevent our model from recognising two word inflections with the same meaning as two entirely separate entities, we use universal tags. In universal tags, all nouns, verbs, adjectives, and adverbs are represented by the tags "N", "V", "A" and "R", respectively. Other words, such as connectives and prepositions receive the same tags as they would in the Treebank format. Once the universal tags have been produced, one can simply concatenate the words in the training corpus with their respective tags. Table 5.3 shows some examples of sentences annotated with universal tags.

Its|||P birthstone|||N is|||V the|||DT diamond|||N .|||.
It|||P has|||V 31|||CD days|||N .|||.
August|||N is|||V named|||V for|||IN Augustus|||N Caesar|||N .|||.
They|||P can|||MD rollback|||V changes|||N .|||.
What|||WP did|||V the|||DT artist|||N value|||N ?|||.

Table 5.3 Sentences annotated with generalised tags

Our new model offers another advantage: the POS tags allow us to augment our Substitution Generation algorithm. By removing all candidates that do not share the same tag as the target word, we prevent our generator from producing certain ungrammatical and/or incoherent substitutions.

Given a set of target words $\{t_1, t_2 ... t_{n-1}, t_n\}$, where each t_i pertains to a sentence S_i $\{S_1, S_2 ... S_{n-1}, S_n\}$, selection limits l and m, and a context-aware embedding model M, the steps of our SG algorithm are:

1. For each target word $t_i \in \{t_1, t_2 ... t_{n-1}, t_n\}$, retrieve universal POS tag P_i by tagging sentence S_i and extracting the POS tag pertaining to t_i.

2. For each target word $t_i \in \{t_1, t_2 ... t_{n-1}, t_n\}$ and universal POS tags $P_i \in \{P_1, P_2 ... P_{n-1}, P_n\}$, extract the l candidates $c_j|||P_j$ from model M which have the shortest cosine distances between $M(t_i|||P_i)$ and $M\left(c_j|||P_j\right)$.

3. Discard any candidate c_j that has the same stem as t_i. Discard any candidate c_j of which the stem is a sub-string of t_i. Discard any candidate c_j which is a sub-string of t_i. Discard any candidate c_j which is a super-string of t_i. Discard any candidate c_j which is a super-string of the stem of t_i. Discard any candidate c_j of which P_j is not identical to P_i.

4. For the remaining candidates of each target word $t_i \in \{t_1, t_2 ... t_{n-1}, t_n\}$, extract the m candidates c_j which have the shortest cosine distances between $M(t_i|||P_i)$ and $M\left(c_j|||P_j\right)$.

5.3 Retrofitting Context-Aware Embedding Models

Mikolov et al. (2013c) shows that one of the linguistic regularities captured by embedding models is synonymy. They noticed that words which have small cosine distances between them are more likely to be equivalent in meaning, and could hence be replaceable in some contexts. They also noticed, however, that antonyms of words are just as likely to have similar embedding vectors. Table 5.4 illustrates this phenomenon. It shows the five words with the highest cosine similarity to "*good*", "*large*", "*thin*", "*short*" and "*blonde*", as determined by an embeddings model trained over 7 billion words, with the CBOW architecture and 500-dimension vectors.

Word	Most Similar
good	bad, decent, really, nice, better
large	small, larger, smaller, sizeable, sizable
thin	thick, slender, thinner, thicker, dense
short	shorter, long, shortened, shortest, shorten
blonde	blond, brunette, dark-haired, blondes, raven-haired

Table 5.4 Most similar words from traditional embedding models

It can be noticed that the antonym of each word is often the most similar in the feature space. This happens because although antonyms have opposite meanings, the contexts in which they are found are often very similar. Since both the CBOW and Skip-Gram models estimate word vectors with respect to the contexts in which they appear in texts, it is expected that these words will indeed have similar embeddings.

This phenomenon can considerably reduce the potential of our SG strategy, since it is very likely that one or more antonyms of a given complex word will be among the generated candidates. Given that the most successful Substitution Selection (Paetzold and Specia, 2015) and Substitution Ranking (Glavaš and Štajner, 2015; Paetzold, 2015b) strategies all rely on n-gram frequencies and embedding cosine distances, allowing for antonyms to be among generated candidates could considerably increase the occurrence of incoherent word replacements.

In order to address this limitation, Faruqui et al. (2015) proposes an algorithm that allows embedding models to be retrofitted over manually created thesauri. Their algorithm modifies the embeddings of words so that they have shorter Euclidean distances to other words with which they share some semantic relation of interest, such as synonymy, hypernymy and hyponymy.

To create such an algorithm, they first take an embeddings model as a matrix $\hat{Q}$, where each column $\hat{q}_i \in \mathbb{R}^d$ is a vector of size d that represents a word w_i in a vocabulary V. In sequence, given a graph (V,E), where each node $(w_i, w_j) \in E \subseteq V \times V$ describes a semantic relation between w_i and w_j, they establish an objective function Ψ to be minimised, which illustrated in Equation 5.4.

$$\Psi(Q) = \sum_{i=1}^{n} \left[\alpha_i \left\| q_i - \hat{q}_i \right\|^2 \sum_{(i,j) \in E} \beta_{ij} \left\| q_i - q_j \right\|^2 \right] \tag{5.4}$$

Algorithm 1 Embeddings Model Retrofitting

```
procedure RETROFIT(Q̂, E, niter)
    Q ← Q̂
    for 0 ≤ k < niter do
        for i ∈ Q do
            related ← j | (i, j) ∈ E
            newvec ← Q(i) * ||related||
            for j ∈ related do
                newvec ← newvec + Q(j)
            end for
            Q(i) ← newvec / (2 * ||related||)
        end for
    end for
    return Q
end procedure
```

In Equation 5.4, Q is an updated (or "retrofitted") version of $\hat{Q}$, and $\Psi(Q)$ its loss. Taking the derivative of Ψ with respect to q_i, they arrive at the update function in Equation 5.5.

$$q_i = \frac{\sum_{j:(i,j)\in E} \beta_{ij} q_j + \alpha_i \hat{q}_i}{\sum_{j:(i,j)\in E} \beta_{ij} + \alpha_i} \tag{5.5}$$

Finally, they translate the update function into Algorithm 1. Applying 100 iterations of Algorithm 1 over the model from which the examples in Table 5.4 were extracted, using the synonymy relations provided by WordNet, we obtain a retrofitted model that captures synonymy more effectively. The five words which are closest to "*good*", "*large*", "*thin*", "*short*" and "*blonde*", as determined by our retrofitted model, are presented in Table 5.5. One can notice that, although the model is still incapable of perfectly capturing synonymy, most antonyms now lie further apart from each other.

Word	Most Similar
good	goodness, beneficial, decent, salutary, honest
large	small, huge, vast, sizeable, sizable
thin	sparse, thick, slender, thinly, slim
short	unawares, shortstop, long, shorter, suddenly
blonde	blond, light-haired, dark-haired, blue-eyed, brown-haired

Table 5.5 Most similar words from retrofitted embedding models

In order for their algorithm to work, the words in model M must be in the same format as the ones in thesaurus T. Consequently, if we were to perform retrofitting over context-aware models, the words in both thesaurus and model would need to be annotated with the same set of POS tags.

To create retrofitted context-aware models, we resort to WordNet. Each sense registered in WordNet can be interpreted as an individual set of synonyms, and is annotated with one of the following universal tags: noun (N), verb (V), adjective (A) or adverb (R), which are the same as the universal tags for content words used by our context-aware models. This means that, if a sense is annotated with the verb tag "V", for example, all registered synonyms are verbs or verbal expressions. To create a set of POS tag annotated synonym relations from WordNet, we iterate through all registered words w_i, and for each of its synonyms w_j, we create an entry $\left(w_i|||p, w_j|||p\right)$, where p is the POS tag of the sense under which w_i and w_j are listed as synonyms. In Table 5.6 are shown some examples of entries.

Once the synonym entries have been produced, one can use Algorithm 1 to retrofit the embeddings of any context-aware model, hence creating a retrofitted context-aware model.

Word	Synonyms
fawn|||V	kowtow|||V, truckle|||V, grovel|||V, suck_up|||V
fawn|||N	dun|||N, grayish_brown|||N, greyish_brown|||N
stern|||N	tail_end|||N, buns|||N, hindquarters|||N, backside|||N
stern|||A	exacting|||A, grim|||A, strict|||A, relentless|||A
travel|||V	journey|||V, go|||V, locomote|||V, trip|||V
travel|||N	travelling|||N, traveling|||N, change_of_location|||N

Table 5.6 POS-annotated lexicon entries

5.4 Experimental Settings

In this Section, we outline the resources and settings used in the experiments conducted with our novel Substitution Generation approach, which employs retrofitted context-aware word embeddings model. We compare our approach with various others, including the state-of-the-art for the task, introduced by Horn et al. (2014). We also perform full pipeline experiments by pairing our generator with various Substitution Selection and Ranking strategies. This way, we evaluate its performance in practice.

5.4.1 Approaches

Substitution Generation

In our experiments, we evaluate the performance of our strategy in four ways: with a typical word embeddings model (TEM), a typical retrofitted model (REM), a context-aware model (CAEM), and a retrofitted context-aware model (RCAEM). All models were trained with the word2vec toolkit[4]. The corpus used contains around 7 billion words, and is composed of all texts extracted from the UMBC webbase, News Crawl, the SUBTLEX corpus, as well as Wikipedia and Simple Wikipedia. The POS tags required for the training of the context-aware model were produced through the use of the Stanford Parser (Klein and Manning, 2003).

We compare our strategy to the following generators:

- **Devlin** (Devlin and Tait, 1998): One of the most frequently used SG strategies, it generates candidate substitutions by extracting all synonyms available in WordNet 3.0 for a target word.
- **Biran** (Biran et al., 2011): Generates candidate substitutions by filtering the cartesian product between the words found in Wikipedia articles and the ones found in the Simple Wikipedia. Their approach first produces all complex-to-simple word pairs between

[4]https://code.google.com/p/word2vec

the Wikipedia and Simple Wikipedia vocabularies, and then applies the following filtering steps over them:

1. Remove all pairs whose vectors in a co-occurrence model have a cosine distance larger than 0.1.
2. Remove all pairs whose words have the same lemma.
3. Remove all pairs where one word is a prefix of the other and its suffix is either "s", "es", "ed", "ly", "er" or "ing".
4. Remove all pairs of which the words are not listed in WordNet as either synonyms or hypernyms.
5. Estimate the complexity of the words in each pair, and then remove all pairs in which the word from Simple Wikipedia is more complex than the one from Wikipedia.

For this approach, we have trained a word co-occurrence model using the same settings used by Biran et al. (2011). The corpora over which the model was trained are the same used in the training of our word embedding models. The function used to determine the complexity of words is also the same one employed by Biran et al. (2011), which is illustrated in Equation 5.6, where $F(w,C)$ is the frequency of word w in corpus C, and $\|w\|$ the length of candidate w.

$$M(w) = \frac{F(w,\text{Complex})}{F(w,\text{Simple})} \times \|w\| \tag{5.6}$$

The "Complex" and "Simple" corpora used by them are Wikipedia and Simple Wikipedia (Kauchak and Barzilay, 2006).

- **Yamamoto** (Kajiwara et al., 2013): Produces candidate substitutions by querying dictionaries for target words, retrieving example and/or definition sentences, and then extracting any words that share the same POS tag as the target word. For this approach, we use the Merriam Dictionary and Thesaurus API [5]. To tag the dictionary examples and definitions, we use the Stanford Parser.

- **Kauchak** (Horn et al., 2014): Generates candidate substitutions from complex-to-simple parallel corpora. Their approach first produces the word alignments from a parallel corpus of sentences extracted from Wikipedia and Simple Wikipedia. It then

[5] http://www.merriam-webster.com

produces a preliminary set of complex-to-simple correspondences by extracting any pairs of aligned content words. Finally, the correspondences are filtered through the following steps:

1. All pairs which do not share the same POS tag are removed.
2. All pairs in which the word from Wikipedia is a stop word are removed.
3. All pairs in which at least one of the words is a proper noun are removed.

The parallel corpus used is the one provided by Kauchak (2013), composed of 150,569 complex-to-simple parallel sentences from Wikipedia and Simple Wikipedia, parsed by the Stanford Parser (Klein and Manning, 2003).

All approaches were replicated with the use of LEXenstein (Paetzold and Specia, 2015). For a more detailed description of the aforementioned approaches, please refer to the literature survey of Chapter 2.

Substitution Selection

In our full pipeline experiments, we pair the aforementioned generators with several Substitution Selection approaches. They are:

- **First Sense (First)**: Assumes that the sense of a given target word is the first one registered in WordNet, and selects as suitable candidates only those words which are listed as synonyms for such sense.
- **Random Sense (Random)**: Selects a random sense from WordNet for a given target word, and selects as suitable candidates only those words which are listed as synonyms for such sense.
- **Lesk Algorithm (Lesk)**: Uses the algorithm introduced by Lesk (1986), frequently referred to in literature as the "Lesk algorithm", to select the word sense in WordNet that best describes a given target word with respect to its context. It does so by selecting the sense with the highest number of overlapping words between the various examples and definitions registered in WordNet for each sense and context of the target word itself.
- **Path Similarity (Path)**: Uses the algorithm introduced by Leacock and Chodorow (1998), which consists on a more sophisticated interpretation of the Lesk algorithm. It takes into account not only the overlap count between a target words' context and

sense examples in WordNet, but also the semantic similarity between the target word and the words in such examples. The semantic similarity between two words a and b is calculated as the distance between them in the WordNet's taxonomy.

- **Word Clustering (Belder)**: Uses a strategy similar to the one presented by De Belder and Moens (2010), in which candidate substitutions are filtered with respect to the classes learned by a latent-variable language model. In our interpretation, we use the algorithm proposed by Brown et al. (1992a), which learns word clusters from large corpora of text. Once the word clusters have been learned, it then finds the cluster in which a given target word is present, and filters any candidates that are not included in said cluster. The corpus used during training is the same one used in the training of the word embedding models used by our SG approach. We have chosen to learn a total of $1,000$ word clusters. We choose this amount of clusters because, to our knowledge, it is the most widely used in literature (Koo et al., 2008; Ratinov and Roth, 2009; Turian et al., 2010).

- **Co-Occurrence Model Filtering (Biran)**: Uses the strategy introduced by Biran et al. (2011), in which candidate substitutions are filtered with respect to the cosine distance between the word co-occurrence vectors of a target word and a candidate substitution. The co-occurrence vector of a target word is composed of the occurrence frequencies of each word present in a 10-token window around it in a given context. The co-occurrence vectors of candidates are learned from their occurrences in large corpora of text. Once the vectors are produced, any candidates of which the cosine distance d between the co-occurrence vector of the target word and its own is either smaller than a lower-bound l or larger than a upper-bound u are filtered out. In our experiments, we use a lower-bound of 0.01 and an upper-bound of 0.1, which are the same used by Biran et al. (2011). The corpus used to obtain the co-occurrence model required is the same one used in the training of the word embedding models used by our SG approach.

Substitution Ranking

For the full pipeline evaluation, we have tested the performance of six SR approaches, which rank candidates according to some metric related to simplicity. The metrics chosen are:

- **Frequency**: Candidates are ranked according to their frequency in the Simple Wikipedia corpus (Kauchak, 2013). The more frequent the word, the simpler it is (Horn et al., 2014).

- **Length**: Candidates are ranked according to their length. We explore the hypothesis that the less characters a candidate have, the simpler it is (Shardlow, 2014a).
- **Senses**: Candidates are ranked according to their number of senses in WordNet. We explore the hypothesis that the more senses a word has, the simpler it is (Shardlow, 2014a).
- **Synonyms**: Candidates are ranked according to their number of synonyms in WordNet. We explore the hypothesis that, much like senses, he more synonyms a word has, the simpler it is Shardlow (2014a).
- **Hypernyms**: Candidates are ranked according to their number of hypernyms in WordNet. The more hypernyms a word has, the simpler it is.
- **Hyponyms**: Candidates are ranked according to their number of hyponyms in WordNet. The more hyponyms a word has, the simpler it is.

We choose these metrics because, as discussed in the survey of Chapter 2, these are some of the most widely used word simplicity metrics.

5.4.2 Datasets

To evaluate the systems, we use the LexMTurk corpus (Horn et al., 2014), which contains 500 instances. Each instance contains a sentence extracted from Wikipedia, a target complex word, and 50 simpler alternatives suggested by turkers from the Amazon Mechanical Turk. We take such suggestions as our gold-standard.

We use the LexMTurk as opposed to NNSeval because it is more than twice as large as NNSeval in both the number of instances and gold candidates. Since we are not attempting to incorporate the needs of non-native English speakers directly during SG, LexMTurk is the most reliable option.

5.4.3 Evaluation Metrics

For the evaluation of Substitution Generation approaches, we use four metrics:

- **Potential**: The proportion of instances in which at least one of the substitutions generated is present in the gold-standard.
- **Precision**: The proportion of generated substitutions that are present in the gold-standard.

- **Recall**: The proportion of gold-standard substitutions that are included in the generated substitutions.
- **F**1: The harmonic mean between Precision and Recall.

The Potential allows for us to evaluate how well the generation approach would perform in an idealised scenario, in which the Substitution Selection designated to filter the generated substitutions is capable of discarding each and every spurious candidate. The Precision, Recall and F1, however, evaluate the systems in a more realistic scenario, in which the results produced by the Substitution Selection approach are not perfect, and hence the LS approach would benefit from generated substitutions of higher quality.

For the full pipeline evaluation, in which various combination of SG, SS and SR approaches are compared, we use Accuracy, which measures the proportion of times in which the candidate substitution ranked first by an SR system is present in the gold-standard and is not the target word itself. This measure is designed to capture how reliable an LS strategy is in replacing target complex words with alternatives that are simpler, yet still grammatical and coherent.

5.5 Comparing Word Embedding Models

Before comparing the performance of our approach with that of other SG strategies, we first conduct a performance assessment using several distinct word embedding models. The models vary in the following aspects:

- **Context-Awareness**: We train both traditional embedding models over raw corpora, and context-aware models over corpora annotated with universal POS tags.
- **Retrofitting**: We train models both with and without the use of retrofitting.
- **Architecture**: We train models with both the CBOW and the Skip-Gram architectures.
- **Size**: We train models with numerical vectors of 300, 500 and 700 dimensions.

By interpolating all possible combinations of settings, we have produced a total of 24 models. Since our approach allows to select how many substitutions to generate, we have evaluated how well each model performs when selecting 5, 10, 15, 20 and 25 substitutions for each target word. The F1 scores obtained by all models evaluated are illustrated in Tables 5.7 through Tables 5.10. The complete results for Potential, Precision and Recall are presented in Tables B.1 through B.16 of Appendix B.

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.111	0.122	0.125	0.099	0.104	0.106
10	0.122	0.138	**0.140**	0.104	0.115	0.115
15	0.122	0.132	0.136	0.101	0.107	0.109
20	0.114	0.124	0.129	0.095	0.100	0.099
25	0.105	0.117	0.120	0.087	0.091	0.092

Table 5.7 F1 results for traditional embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.121	0.129	0.132	0.115	0.123	0.127
10	0.137	0.150	**0.152**	0.128	0.138	0.141
15	0.134	0.145	0.148	0.125	0.134	0.134
20	0.125	0.137	0.140	0.117	0.123	0.125
25	0.119	0.131	0.133	0.109	0.114	0.115

Table 5.8 F1 results for retrofitted embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.127	0.140	0.146	0.113	0.125	0.128
10	0.141	0.156	**0.164**	0.129	0.138	0.140
15	0.142	0.153	0.159	0.128	0.138	0.138
20	0.135	0.144	0.151	0.123	0.132	0.134
25	0.127	0.138	0.143	0.120	0.128	0.128

Table 5.9 F1 results for context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.144	0.150	0.154	0.135	0.141	0.141
10	0.159	0.170	**0.175**	0.151	0.160	0.159
15	0.156	0.166	0.171	0.149	0.157	0.157
20	0.149	0.159	0.163	0.143	0.149	0.150
25	0.141	0.150	0.154	0.137	0.144	0.145

Table 5.10 F1 results for retrofitted context-aware embedding models

Several conclusions can be drawn from the results obtained. The CBOW model is more effective than the the Skip-Gram model for the task, and using larger numerical vectors can considerably increase the performance of our approach. It can also be noticed that retrieving more candidate substitutions increases Potential and Recall, but decreases Precision, which consequently makes it challenging for the optimal amount of candidates retrieved to be found.

Perhaps the most interesting phenomenon observed refers to the effectiveness of context-aware embedding models for SG: they have outperformed traditional and retrofitted models in Potential, Precision, Recall and F1 in every configuration evaluated. Combining retrofitting with context-aware models yield even better results. Such observations highlight the potential inherent to such models, and allow us to conclude that they, at least for the purposes of Lexical Simplification, produce numerical vector representations more reliable than those of a traditional embedding models.

We also assess the performance of our generator in its four configurations (TEM, REM, CAEM and RCAEM) when using word vectors sizes that go beyond the 300, 500 and 700 dimensions featured in the previous assessment. Figure 5.4 illustrates the scores obtained with vectors of 500 to $2,500$ dimensions, in intervals of 200. All models use the CBOW architecture, and select 10 candidates for each target word.

One can notice that the performance of all models grow in the same pattern, with a noticeable drop at $1,500$ dimensions and a steady increase thereafter. Assessing the substitutions produced by the models, we found that the main reason for the performance drop lies in the fact that the proposed models do not discern between the meanings of a word within the same grammatical class. Take the target word "*site*" in its Noun form, for an example. While among the candidates produced by the models with $1,300$ dimensions were words which referred "*site*" as both an outlined portion of terrain and a webpage, such as "*location*" and "*homepage*", the ones produced by the model with $1,500$ dimensions referred strictly to "*site*" as a webpage, which happened not to be the required sense in question. This phenomena suggests that, although going beyond the usual 300 or 500 dimensions can benefit the performance of our generators, using excessively high dimensions can lead the model to approximate an ambiguous word to its most frequently used sense within a given grammatical class, limiting its efficacy in SG.

5.6 Comparing Generation Approaches

In this experiment, we compare ours to several other approaches to SG. For the purpose of analysis, we include all versions of our approach: when using a traditional embeddings

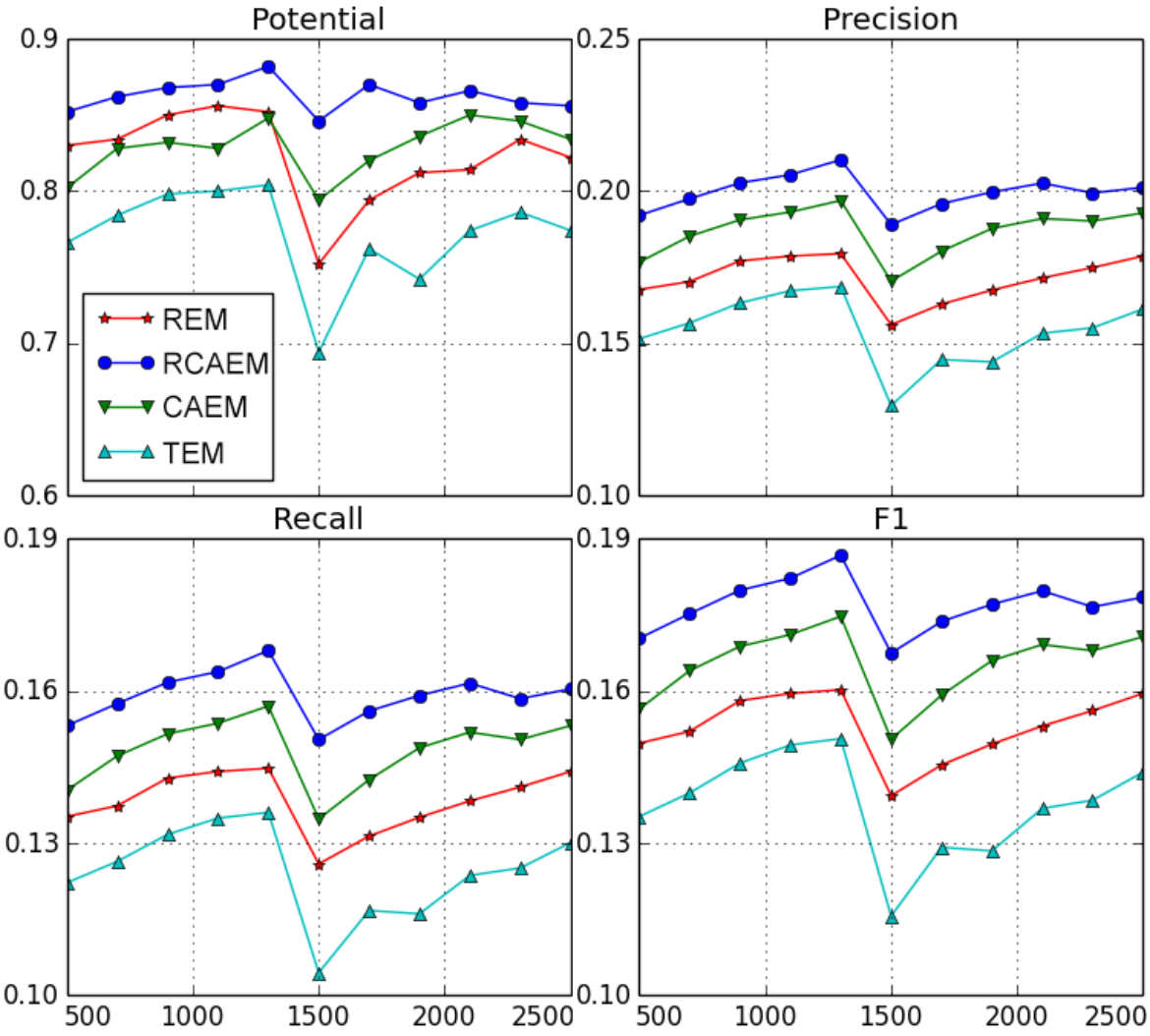

Fig. 5.4 Vector size versus SG scores

model (TEM), a retrofitted model (REM), a context-aware model (CAEM) and a retrofitted context-aware model (RCAEM). All versions of our approach retrieve a total 10 candidate substitutions for each target word. The models used by both versions were trained with the CBOW architecture, and use numerical vectors with $1,300$ dimensions. The results obtained are shown in Table 5.11.

Approach	Potential	Precision	Recall	F1
Biran	0.630	0.153	0.098	0.119
Kauchak	0.784	**0.290**	0.115	0.164
Devlin	0.616	0.157	0.088	0.113
Yamamoto	0.510	0.056	0.079	0.065
TEM	0.804	0.169	0.136	0.151
REM	0.852	0.179	0.145	0.160
CAEM	0.848	0.197	0.157	0.175
RCAEM	**0.882**	0.210	**0.168**	**0.187**

Table 5.11 Performance scores for various SG approaches

When using a retrofitted context-aware model, our SG approach was able to outperform all other strategies evaluated, achieving the highest scores for Potential, Recall and F1. Although the Kauchak generator has managed to outperform our approach in Precision, the strategy it uses obtains a noticeably lower Recall and F1, despite relying on manually produced data. The performance obtained with a traditional embeddings model is nonetheless quite impressive: it outperformed almost all baseline systems in Potential, Recall and F1, even though it is a completely unsupervised SG approach.

Although our approach manages to obtain only a 2.3% improvement in F1 over the second best generator evaluated, this difference could still imply in major gains in performance when these generators are incorporated in an LS system. Evidence of that can be found in the experiments of Horn et al. (2014), who compare the performance of their simplifier, which uses the Kauchak generator, with that of Biran et al. (2011), which uses the Biran generator. Even though the difference in F1 between these generators is only 4.5%, the simplifier of Horn et al. (2014) is 42.7% more accurate than the simplifier of Biran et al. (2011) when employed in practice.

Those results confirm the advantage of context-aware over typical embedding models in producing meaningful numerical vector representations of words, and reveal that ours is a state-of-the-art approach for Substitution Generation.

5.7 Full Pipeline Performance Comparison

To evaluate how well our generators perform in practice, we assess their performance when introduced in a complete LS pipeline. We first pair them with the various Substitution Selection approaches described in Section 5.4.1. The results obtained are presented in Tables 5.12 through 5.17.

Selector	Potential	Precision	Recall	F1
Biran	0.064	0.157	0.005	0.010
Kauchak	0.052	0.127	0.004	0.008
Devlin	**0.098**	0.141	**0.010**	**0.018**
Yamamoto	0.014	0.036	0.001	0.002
TEM	0.062	0.122	0.005	0.011
REM	0.084	0.181	0.009	0.017
CAEM	0.072	0.158	0.006	0.012
RCAEM	0.080	**0.194**	0.008	0.015

Table 5.12 Results obtained by pairing different generators with the Random Sense approach (Random)

Selector	Potential	Precision	Recall	F1
Biran	0.116	0.239	0.011	0.021
Kauchak	0.072	0.176	0.006	0.012
Devlin	**0.162**	0.204	**0.017**	**0.032**
Yamamoto	0.038	0.079	0.003	0.006
TEM	0.122	0.197	0.011	0.022
REM	0.148	0.236	0.015	0.029
CAEM	0.132	0.256	0.012	0.024
RCAEM	0.152	**0.278**	0.015	0.029

Table 5.13 Results obtained by pairing different generators with the First Sense approach (First)

Although the Devlin generator has obtained the highest scores when paired with First and Random Sense selectors, these baselines performed very poorly overall, since they do not perform any type of intelligent selection procedure. For the remaining selectors, however, our SG approach has managed to outperform all other generators. Even though the highest overall scores were obtained by the RCAEM generator, it can be noticed that the performance of each type of embeddings model varies according to the selector used.

We have also evaluated the performance of each and every SG and SS combination when paired with several metric-based Substitution Ranking approaches. The Accuracy scores obtained are shown in Tables 5.18 through 5.23.

Selector	Potential	Precision	Recall	F1
Biran	0.100	0.100	0.009	0.016
Kauchak	0.088	0.179	0.010	0.019
Devlin	0.140	0.096	0.014	0.024
Yamamoto	0.038	0.031	0.003	0.006
TEM	0.222	0.135	0.024	0.041
REM	0.236	0.148	0.028	0.047
CAEM	0.232	0.164	0.026	0.046
RCAEM	**0.238**	**0.184**	**0.030**	**0.052**

Table 5.14 Results obtained by pairing different generators with the Lesk Algorithm (Lesk)

Selector	Potential	Precision	Recall	F1
Biran	0.016	0.065	0.001	0.003
Kauchak	0.012	0.034	0.001	0.002
Devlin	0.026	0.093	0.003	0.006
Yamamoto	0.004	0.010	0.000	0.001
TEM	0.028	0.060	0.002	0.004
REM	0.032	0.116	0.003	0.007
CAEM	0.032	0.095	0.003	0.005
RCAEM	**0.032**	**0.165**	**0.003**	**0.007**

Table 5.15 Results obtained by pairing different generators with the Path Similarity approach (Path)

Selector	Potential	Precision	Recall	F1
Biran	0.124	0.482	0.011	0.021
Kauchak	0.228	**0.585**	0.022	0.043
Devlin	0.120	0.447	0.011	0.022
Yamamoto	0.080	0.286	0.007	0.014
TEM	0.290	0.340	0.030	0.055
REM	0.280	0.352	0.029	0.053
CAEM	**0.330**	0.345	**0.035**	**0.063**
RCAEM	0.312	0.375	0.033	0.061

Table 5.16 Results obtained by pairing different generators with the Word Clustering approach (Clusters)

Selector	Potential	Precision	Recall	F1
Biran	0.366	0.112	0.057	0.075
Kauchak	0.450	**0.298**	0.066	0.108
Devlin	0.360	0.127	0.050	0.072
Yamamoto	0.318	0.061	0.047	0.053
TEM	0.480	0.175	0.079	0.109
REM	0.492	0.184	0.082	0.113
CAEM	0.502	0.200	0.088	0.122
RCAEM	**0.520**	0.217	**0.096**	**0.133**

Table 5.17 Results obtained by pairing different generators with the Co-Occurrence Model Filtering approach (Biran)

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.086	0.054	0.016	0.066	0.116	0.110
Kauchak	0.040	0.024	0.008	0.064	0.216	0.150
Devlin	0.112	**0.078**	0.026	0.086	0.114	0.124
Yamamoto	0.028	0.010	0.002	0.010	0.070	0.042
TEM	0.110	0.056	0.022	0.128	0.228	0.186
REM	0.116	0.072	**0.030**	0.124	0.224	0.180
CAEM	0.116	0.068	**0.030**	**0.146**	**0.252**	**0.208**
RCAEM	**0.120**	0.072	**0.030**	0.140	0.242	0.206

Table 5.18 Accuracy scores for candidate substitutions ranked by their length

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.104	0.060	0.016	0.082	0.118	0.150
Kauchak	0.076	0.034	0.012	0.074	0.222	0.228
Devlin	**0.150**	**0.092**	0.024	0.110	0.114	0.196
Yamamoto	0.030	0.014	0.002	0.018	0.070	0.078
TEM	0.110	0.058	0.024	0.128	0.246	0.240
REM	0.136	0.078	0.026	0.138	0.234	0.254
CAEM	0.126	0.072	**0.030**	0.152	**0.270**	0.264
RCAEM	0.140	0.078	0.028	**0.156**	0.264	**0.278**

Table 5.19 Accuracy scores for candidate substitutions ranked by their frequency

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.102	0.060	0.016	0.080	0.122	0.144
Kauchak	0.076	0.034	0.012	0.076	0.214	**0.276**
Devlin	**0.132**	**0.078**	0.020	0.100	0.114	0.152
Yamamoto	0.036	0.014	0.002	0.020	0.068	0.068
TEM	0.102	0.050	0.022	0.120	0.230	0.184
REM	0.124	0.066	0.024	0.126	0.222	0.196
CAEM	0.110	0.072	**0.030**	**0.146**	**0.256**	0.206
RCAEM	0.126	0.070	0.026	**0.146**	0.250	0.232

Table 5.20 Accuracy scores for candidate substitutions ranked by their number of senses

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.102	0.058	0.016	0.072	0.118	0.114
Kauchak	0.076	0.034	0.012	0.076	0.220	**0.266**
Devlin	**0.136**	**0.076**	0.020	0.094	0.112	0.130
Yamamoto	0.036	0.014	0.002	0.020	0.072	0.056
TEM	0.104	0.050	0.022	0.122	0.232	0.176
REM	0.122	0.064	0.024	0.114	0.218	0.182
CAEM	0.112	0.072	**0.030**	**0.150**	**0.264**	0.212
RCAEM	0.128	0.068	0.026	0.142	0.250	0.218

Table 5.21 Accuracy scores for candidate substitutions ranked by their number of synonyms

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.100	0.058	0.016	0.070	0.118	0.108
Kauchak	0.072	0.034	0.012	0.072	0.210	**0.258**
Devlin	**0.128**	**0.074**	0.020	0.084	0.114	0.122
Yamamoto	0.036	0.014	0.002	0.016	0.068	0.062
TEM	0.092	0.044	0.016	0.114	0.218	0.162
REM	0.116	0.066	**0.024**	0.120	0.212	0.176
CAEM	0.102	0.064	**0.024**	**0.136**	**0.238**	0.190
RCAEM	0.124	0.066	0.026	**0.136**	0.234	0.208

Table 5.22 Accuracy scores for candidate substitutions ranked by their number of hypernyms

	First	Random	Path	Lesk	Clusters	Biran
Biran	0.100	0.060	0.016	0.076	0.114	0.096
Kauchak	0.072	0.034	0.012	0.074	0.204	**0.262**
Devlin	**0.130**	**0.076**	0.020	0.096	0.112	0.128
Yamamoto	0.036	0.014	0.002	0.020	0.072	0.044
TEM	0.094	0.044	0.016	0.100	0.210	0.190
REM	0.116	0.064	0.024	0.108	0.206	0.188
CAEM	0.104	0.062	0.024	0.122	**0.234**	0.198
RCAEM	0.126	0.066	**0.026**	**0.126**	0.230	0.204

Table 5.23 Accuracy scores for candidate substitutions ranked by their number of hyponyms

An interesting phenomenon can be observed in the results: pairing the Kauchak generator with Co-Occurrence Model Filtering proved to be effective when the SR approach used ranks candidates with a metric related to ambiguity i.e. number of senses, synonyms and hypernyms. This observation serves as evidence that, in a pipelined setup, it is difficult to predict which configuration will yield the best results, and hence experimenting on a development set with various combinations of strategies is very important if the performance of an LS system is to be maximised. Nonetheless, pairing the RCAEM generator with Co-Occurrence Model Filtering and frequency ranking yields the best Accuracy scores in practice, which asserts the effectiveness of retrofitted context-aware models in SG.

5.8 Conclusions

In this Chapter we describe a new approach for Substitution Generation, which generates candidate substitutions by using the semantic relations found in word embedding models. We also introduce the concept of retrofitted context-aware embedding models, which not only assign a distinct numerical vector to every grammatical form that the word may take, but also exploit synonymy relations in manually created lexicons.

In the experiments conducted, our SG approach have managed to outperform several other generators, not only in the task of Substitution Generation alone, but also when paired with different SS and SR approaches to create complete LS strategies. We have found that, although context-aware models are significantly more effective than typical models in retrieving meaningful candidate substitutions, retrofitting them over WordNet can further increase their reliability. We also found that selecting the appropriate size for the embeddings used can greatly influence the performance of our SG approach.

Given the considerations made, we can conclude that ours is an effective approach to Substitution Generation. It can outperform even generators that rely on scarce linguistic

resources, and offers both flexibility and extensibility. It allows for one to choose how many candidates to be retrieved, and hence control the balance between Precision and Recall, and it can also be easily adapted to various languages. In the case of languages such as English, for which resources are abundant, one can generate candidates with the more sophisticated retrofitted context-aware models, which require both a reliable POS tagger and a thesaurus. As for languages for which those resources are not available, one can still perform generation by using traditional embedding models, which require only for large corpora of text.

Chapter 6

Supervised Models for Substitution Ranking

As discussed in the survey of Chapter 2, Substitution Ranking (SR) is the task of, given a set of substitution candidates for a complex word in a sentence, ranking them in order of simplicity. The final goal of the task is to allow for an LS approach to select the simplest candidate available for a given substitution.

Although easy to grasp, SR can be a very challenging task, since its very foundation lies on attempting to model the abstract concept of "simplicity". Simplicity is often defined as a quality pertaining to content which is easy to understand, given the needs of a certain target audience. For the aphasic, for example, simple sentences can be those which are short and do not contain segments in passive voice (Devlin, 1999), while for non-native speakers, sentences that contain familiar words tend to be simpler (Nation, 2001; Nation and Coady, 1988).

Due to the nature of the LS task and the target audience being addressed (non-native speakers of English), we interpret simplicity as word familiarity i.e. the more familiar a word is to the reader, the simpler it is. In this Section we report on our attempt to conceive a new supervised strategy for Substitution Ranking that accounts for the needs of non-native English speakers. Our approach, which we name Boundary Ranking, uses weighted linear models trained over a binary classification setup to learn a ranking model from the needs of a target audience.

Notice that, although Substitution Ranking is performed after Substitution Selection in the typical LS pipeline, we address it first in this thesis. The reason for that lies in the fact that our Substitution Selection approach employs the supervised ranking approach described in this Chapter.

6.1 Reinterpreting the Task: What Makes a Good Ranker?

In order to introduce a new approach to SR, it is important to first understand what the intrinsic characteristics of an effective candidate ranker are in practice.

Through the literature survey described in Chapter 2, we were able to notice that the evaluation metric most frequently used to measure the performance of SR approaches is the Kappa agreement coefficient. This is one of the metrics used to compare the systems submitted to the English Lexical Simplification task of SemEval 2012 (Specia et al., 2012), and it measures the agreement proportion between the judgments made by a system and those made by human annotators (Carletta, 1996). The Kappa agreement coefficient can be calculated as described in Equation 6.1.

$$K = \frac{P(A) - P(E)}{1 - P(E)} \tag{6.1}$$

In Equation 6.1 $P(A)$ represents the agreement proportion between a ranker and a gold-standard, and $P(E)$ the probability of them agreeing by chance. Note that, given the nature of the task, $P(A)$ and $P(E)$ can be calculated in many different ways. In ranking tasks, the pairwise Kappa variant is used: it estimates $P(A)$ as the proportion of agreements between a ranking system and the gold-standard with respect to a pair of candidates. In simpler terms, if both ranker and gold-standard determine that the ranking position of a candidate c_i is higher ($>$), lower ($<$) or equal ($=$) to the one of candidate c_j, the count of agreements is increased by one. This verification is repeated to all possible pairs of ranked candidates $\left(c_i, c_j\right)$ such that $i \neq j$.

This metric is the most popular choice among those who have published work on Substitution Ranking after SemEval 2012, such as in (Kauchak, 2013), (Shardlow, 2013b) and (Glavaš and Štajner, 2015). The fact that most choose to use the Kappa coefficient to evaluate the efficiency of their approaches serves as evidence that, within the LS research community, it is generally accepted that the best SR approach available is the one capable of producing a ranking most similar to a gold-standard. In other words, this metric reflects the ability of systems to rank all candidate substitutions as a human would. We, however, believe that this not the best way to assess the quality of Substitution Ranking systems in practice.

The reason why we disagree with that hypothesis is that, although the pairwise Kappa coefficient is an interesting way to measure the similarity between two rankings, it does not necessarily capture the potential of an SR approach to be successful in contributing to the practical goal of an LS system, which is to replace a given complex word with a simpler alternative. In other words, differences in the ranks of the words that are not ranked first are not important in practice. Consider, for example, the task of replacing the word "*endorsed*"

in sentence "*Kowal suggested the name and the IAU endorsed it in 1975.*". Let's suppose tthat of the most effective supervised and unsupervised rankersat there are five candidate substitutions available, ranked in order of simplicity, unanimously, by members of a certain target audience in the following fashion:

1. approved
2. supported
3. adopted
4. okayed
5. backed

Now let's consider two ranking systems $S1$ and $S2$, which rank the same candidates in the following way:

- **S1**: 1: approved, 2: backed, 3: supported, 4: okayed, 5: adopted
- **S2**: 1: okayed, 2: approved, 3: supported, 4: adopted, 5: backed

At first glance, one may assume that $S1$ have made more mistakes than $S2$, since more candidates seem to be "out of place" with respect to the gold-standard. When estimating the value of $P(A)$ of the pairwise Kappa agreement coefficient for both systems, we obtain a score of 6 for $S1$ and 7 for $S2$, which is in accordance with our initial assumption, meaning that that $S2$ is a better ranker than $S1$. In a practical context, however, an LS approach which uses $S1$ will be considerably more effective than one which uses $S2$, since $S1$ was capable of placing the candidate substitution deemed most simple in the first ranking position.

Based on this observation, we conclude that, in practice, it is more effective to evaluate a Substitution Ranking system based on the quality of its highest ranking substitution, rather than on the overall similarity between its ranking and a gold-standard. Intuitively, we can also propose that it is more effective to train a ranker which maximises the likelihood of ranking "good" candidates first, rather than one which maximises its correlation with a gold-standard. The supervised SVM ranking approach of Joachims (2002) is, to our knowledge, the only example of supervised ranker used in LS, and it happens to suffer from this limitation. Although the results obtained by Jauhar and Specia (2012) and Horn et al. (2014) show that this approach is promising for SR, these SVMs are trained with the goal of minimizing the overall number of errors made by the ranker, which we believe could limit their potential in LS. It is important to mention that, although supervised approaches require manually

annotated data, they are the only ones that allow for a model of the needs of a target audience to be learned automatically. Because of that, we believe they are the most suitable alternative when it comes to addressing the needs of non-native English speakers.

In the Sections that follow, we describe a new supervised strategy for the task, which explores the aforementioned hypothesis by proposing a new way of modelling Substitution Ranking (Section 6.2) and conduct several experiments with it (Sections 6.4 through 6.6).

6.2 Boundary Ranking: A Supervised Approach

Based on the discussion in the previous Section we outline what best characterises an effective Substitution Ranker: it is not its ability to order candidates with respect to their simplicity, but its proficiency in placing simpler candidates in the highest ranking. In order to create a novel strategy for Substitution Ranking that exploits this observation, we first propose a binary classification setup for the task.

Suppose that we are given a set of ranking training examples in the format illustrated in Equation 6.2, where S_i is the ith sentence in a dataset, w_i a target complex word in the h_ith position of S_i, c_i^j a substitution candidate and r_i^j its simplicity ranking. The LexMTurk (Horn et al., 2014) and LSeval (De Belder and Moens, 2012b) corpora are some examples of LS datasets that contain this type of information. Notice that these datasets are a way of representing the simplification needs of a certain target audience, given that the substitution candidates and their simplicity rankings can be produced by human annotators of said target audience.

$$\begin{bmatrix} \langle \mathrm{S}_1 \rangle \ \langle \mathrm{w}_1 \rangle \ \langle \mathrm{h}_1 \rangle \ \langle \mathrm{r}_1^1{:}\mathrm{c}_1^1 \rangle \cdots \langle \mathrm{r}_1^n{:}\mathrm{c}_1^n \rangle \\ \langle \mathrm{S}_2 \rangle \ \langle \mathrm{w}_2 \rangle \ \langle \mathrm{h}_2 \rangle \ \langle \mathrm{r}_2^1{:}\mathrm{c}_2^1 \rangle \cdots \langle \mathrm{r}_2^n{:}\mathrm{c}_2^n \rangle \\ \vdots \\ \langle \mathrm{S}_{m-1} \rangle \ \langle \mathrm{w}_{m-1} \rangle \ \langle \mathrm{h}_{m-1} \rangle \ \langle \mathrm{r}_{m-1}^1{:}\mathrm{c}_{m-1}^1 \rangle \cdots \langle \mathrm{r}_{m-1}^n{:}\mathrm{c}_{m-1}^n \rangle \\ \langle \mathrm{S}_m \rangle \ \langle \mathrm{w}_m \rangle \ \langle \mathrm{h}_m \rangle \ \langle \mathrm{r}_m^1{:}\mathrm{c}_m^1 \rangle \cdots \langle \mathrm{r}_m^n{:}\mathrm{c}_m^n \rangle \end{bmatrix} \tag{6.2}$$

Over each instance of such data, we can propose a function $\mathbf{x} = f\left(\mathrm{c}_i^j\right)$, which represents a set of feature values that describe candidate c_i^j with respect to target complex word w_i and sentence S_i. We can also assign a binary label y to each candidate c_i^j, which takes value 1 if $\mathrm{r}_i^j \leq p$, and 0 otherwise. The parameter p represents the maximum ranking position that a candidate c_i^j can have in order for it to be considered simple enough to replace a target word w_i. This parameter grants more flexibility to the setup: one can train more permissive ranking

models that better fit the data of training sets that contain larger amounts of candidates, for example.

Once a set of training feature values **x** and binary labels *y* are collected, we can then train a linear or non-linear model over the data. To do so, we can use various learning methods to train a model that separates candidates that are simple enough to simplify a target complex word from those that are not. We can then use the model to predict if an unseen candidate is simple enough to replace a target complex word. The predicted binary label however, would not allow for us to decide which is the simplest candidate in the case that more than one candidate from a given set receive label 1. To solve that problem, we can instead use the confidence estimates produced by the model as a measure of how simple each candidate is, which fulfills our setup's final goal and allows us to rank all candidates by simplicity.

In order to produce confidence estimates, weighted linear models and Support Vector Machines, for example, estimate a hyperplane between the two classes in a binary classification problem. Once the optimal hyperplane is found, these confidence estimates can be calculated as the distance between the hyperplane and the feature values that represent a certain word being ranked. The main intuition behind this approach is that the hyperplane learned by the model will be perpendicular to the direction in which the properties being captured by the features grow. In this scenario, the positive and negative binary classification instances used for training would serve as the needle of a "compass" that determines the direction in which the properties being targeted grow. Notice that, although this approach was conceived for the purpose of word simplicity ranking, it could be adapted to different ranking tasks and domains through the use of different datasets and features.

We name our approach **Boundary Ranking**, in reference to the "boundary" (or hyperplane) between positive and negative training samples that weighted linear and non-linear models learn from the binary classification setup. With our supervised SR approach formalised, only one engineering step remains missing: deciding which settings and features should be used so that our ranking model produces optimal results. In what follows, we describe our efforts in finding the most suitable set of configurations for our ranking model.

6.3 Experimental Settings

In this Section, we outline the resources and settings used in the experiments conducted with our novel Substitution Ranking approach. We compare our approach in various settings with current state-of-the-art rankers.

6.3.1 Datasets

In order for us to be able to compare our results with the state-of-the-art for the task, we have chosen to train and test our approach in the dataset from the English Lexical Simplification task of SemEval 2012, which contains 300 training instances and 1,710 test instances. Each instance is composed of a sentence, a target word to be simplified, and a list of candidate substitutions with a reference ranking determined by non-native speakers of the English language. We choose this dataset because it was created by the target audience being addressed in this thesis, and is the largest and most widely used SR dataset in literature.

6.3.2 Evaluation Metrics

In Section 6.1, we have discussed that the evaluation metric most frequently used to measure the performance of Substitution Ranking approaches does not reliably represent the effectiveness of a ranker in the context of LS. It consequently would not be sensible to conceive a strategy that exploits a different criterion for training, but evaluate it with a metric that does not capture said criterion.

In order to address this problem, we present more sensible evaluation metrics for SR. If the effectiveness of a ranking approach is determined by its ability to place simpler substitutions in the highest ranking position, then intuitively its performance should be measured with respect to the simplicity of its highest ranking substitution. To represent that, we employ the TRank-at-n metric, that, given a set of test instances with reference rankings, measures the proportion of times in which a ranking approach places a substitution candidate of reference rank $r \leq n$ in first place.

The TRank-at-1 measure was first introduced by Specia et al. (2012), where it is referred to simply as TRank, and was used to evaluate the approaches submitted to SemEval 2012. By introducing the n range, we allow more flexibility.

6.3.3 Features

We consider a set of 26 features to train our ranking strategy. They can be divided in four categories:

- **Lexicon-oriented**: Features that exploit the occurrence of a given candidate in a lexicon of words or collection of content. They can be discrete and estimate the number of documents or sentences in which a candidate appears given a corpus, or binary and receive value 1 if a candidate appears in a given vocabulary, and 0 otherwise. We include four features of this category, two discrete and two binary. The discrete

features calculate the number of documents in which a word appears, and considers two document collections: the articles from Simple Wikipedia (Kauchak, 2013), and the subtitles from the SubIMDB corpus, introduced in Section 8.1. The binary features consider two vocabularies: one extracted from Simple Wikipedia articles, and another from Ogden's Basic English lexicon[1].

- **Morphological**: Features that exploit morphological characteristics of candidate substitutions. We include two features of this category: word length and number of syllables. To split words into syllables, we resort to Morph Adorner (Burns, 2013).

- **Semantic**: Features which are related to the meaning of a candidate substitution. We include seven features of this category: number of senses, synonyms, hypernyms, hyponyms and maximum and minimum distances between a word's sense and the most general sense in a thesaurus. We use WordNet to obtain these features.

- **Collocational**: N-gram probabilities of the form $P\left(S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}\right)$, where c is a candidate substitution replacing a target word in the hth position of sentence S, and S_{h-l}^{h-1} and S_{h+1}^{h+r} are n-grams of size l and r, respectively. We include 16 features of this category, which are the frequency of each candidate alone, as well as all n-grams with windows $l \leq 3$ and $r \leq 3$ to the left and right of each candidate substitution in position h of S. To obtain n-gram probabilities, we train a 3-gram language model over SubIMDB using SRILM (Stolcke, 2002).

6.3.4 Baselines

In the experiment of Section 6.6, we compare the performance of Boundary Ranking with that of the most effective supervised and unsupervised rankers in literature:

- **Google 1T:** Ranks candidates according to their frequency in the Google 1T corpus. Although simple, this approach obtained the second highest TRank-at-1 scores in the English Lexical Simplification task of SemEval 2012.

- **UOW-SHEF (Jauhar and Specia, 2012):** Uses a linear weighted function over rankings produced by various collocational, morphological and psycholinguistic features. This approach obtained the highest TRank-at-1 scores in the English Lexical Simplification task of SemEval 2012.

[1] http://ogden.basic-english.org

- **Glavas (Glavaš and Štajner, 2015):** An unsupervised approach that ranks candidates according to their average ranking positions, as determined by a given set of features. The experiments of Glavaš and Štajner (2015) show that their approach achieves Kappa correlation scores much higher than that of the best system submitted to the English Lexical Simplification task of SemEval 2012.

- **SVMRank (Joachims, 2002):** Learns a supervised ranking model with Support Vector Machines. This approach was used in the creation of a state-of-the-art LS system by Horn et al. (2014). Unlike Boundary Ranking, this model does not prioritize the quality of the candidate with the highest ranking position, but rather attempts to maximize the correlation between predicted and gold-standard rankings. To our knowledge, this is the only supervised ranking strategy available for LS apart from Boundary Ranking.

6.4 Training Settings Assessment

As discussed in Section 6.2, Boundary Ranking consists in estimating the simplicity of each candidate in a set of substitutions by measuring how confident a model is that said candidate is simple enough to replace a target complex word. In weighted linear and non-linear models, this is calculated as the projected distance between a candidate's feature vector and the estimated hyperplane.

In linear models, estimating a hyperplane is usually done through learning algorithms that minimise a loss function with respect to a regularisation term, and in non-linear models, through Support Vector Machines with non-linear kernels. In this experiment, we evaluate how these different settings influence the effectiveness of our Boundary Ranking approach.

6.4.1 Linear Models

One can find various examples of loss functions and regularisation terms for linear models in literature. For our experiments, we have selected three of the most widely used loss functions in literature:

- **Hinge**: A loss function commonly used for the training of soft-margin linear Support Vector Machines, the Hinge loss l of a weight vector $\mathbf{w}$ with respect to a training set of n examples, each represented by a label $y_i \, \varepsilon \, \{1, -1\}$, can be simply defined as:

$$l(\mathbf{w}) = \sum_{i=1}^{n} \max\left(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right) \tag{6.3}$$

- **Squared Hinge**: The Squared Hinge loss is a variation of the Hinge loss that penalises errors quadratically. It can be defined as:

$$l(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n} \max\left(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right)^2 \tag{6.4}$$

- **Modified Huber**: It is an adaptation of the Huber loss (Huber, 1964) for classification tasks. The Huber loss was designed to be less sensitive to outliers. It can be defined as:

$$l(\mathbf{w}, \mathbf{x}_i, y_i) = \begin{cases} \max\left(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right) & \text{if } y_i \mathbf{w}^T \mathbf{x}_i \geq -1 \\ -4 y_i \mathbf{w}^T \mathbf{x}_i & \text{otherwise} \end{cases} \tag{6.5}$$

During learning, loss functions are often summed to a regularisation term. The goal of using a regularisation term during learning is to prevent the model from overfitting the data and hence generalising poorly for unseen examples. We have selected three of the most widely used regularisation terms in literature to test with each loss function:

- **L1**: Represents the sum of the absolute values of all weights w_i in weight vector $\mathbf{w}$. It is described in Equation 6.6, where α is a constant that scales the regularisation term when summed to a loss function.

$$L_1(\mathbf{w}) = \alpha \sum_{i=1}^{m} |w_i| \tag{6.6}$$

  L**1** regularisation has been shown to be more effective for scenarios where the data is characterised by high sparsity (Shalev-Shwartz and Tewari, 2009).

- **L2**: It is similar to L**1** regularisation, and represents the sum of the squared values of all weights w_i in weight vector $\mathbf{w}$. It is illustrated in Equation 6.7.

$$L_2(\mathbf{w}) = \alpha \sum_{i=1}^{m} (w_i)^2 \tag{6.7}$$

  L**2** regularisation has been shown to be more effective in scenarios where the data is characterised by low sparsity (Moore and DeNero, 2011).

- **Elastic Net**: First introduced by Zou and Hastie (2005), the elastic net linearly combines L**1** and L**2**. It is illustrated in Equation 6.8, where λ_1 is the scale of regularisation term $L_1(\mathbf{w})$, and λ_2 is the scale of regularisation term $L_2(\mathbf{w})$.

$$L_e(\mathbf{w}) = \lambda_1 L_1(\mathbf{w}) + \lambda_2 L_2(\mathbf{w}) \tag{6.8}$$

Elastic Net regularisation was devised in an effort to address some limitations of L**1** and L**2**, and have been shown to outperform both of them in scenarios where the training data is high dimensional (Waldron et al., 2011).

There are also various algorithms that can be used to learn the model parameters over the training data. Each learning algorithm has its own set of hyper-parameters, and some of them do not support all the aforementioned loss functions and regularisation terms. We have selected two of the most widely used learning algorithms in literature for our experiments:

- **Stochastic Gradient Descent**: An iterative gradient descent optimisation method used for stochastic linear models, or in other words, linear models whose behavior cannot be predicted in a deterministic fashion. It uses an iterative batch learning setup, where its goal is to improve on the overall classification performance of the estimated hyperplane after each iteration with respect to a regularised loss calculated over a random sample (or "batch") of training instances. Formally, each iteration of the Stochastic Gradient Descent algorithm takes the form illustrated in Equation 6.9, where w_t is the weight vector produced in iteration $t-1$, w_{t+1} the resulting incremented weight vector of iteration t, η_t a learning rate, and $\triangledown Q(w_t)$ vector of partial derivatives (or gradient) of $Q(w_t, X_t, Y_t)$, which represents a regularised loss function calculated over the current weight vector w_t and a batch of training samples X_t with labels Y_t.

$$w_{t+1} = w_t - \eta_t \triangledown Q(w_t, X_t, Y_t) \tag{6.9}$$

  Stochastic Gradient Descent supports any differentiable convex loss function, and can hence use all the aforementioned loss functions and regularisation terms.

- **Passive Aggressive Learning**: An online algorithm first introduced by Crammer et al. (2006), it iteratively increments vector **w** based on the classification error of a training instance. Its goal is to improve on the overall classification performance of the estimated hyperplane after each iteration, while ensuring that the knowledge acquired in previous iterations is not lost. Formally, each iteration of the Passive Agressive algorithm takes the form illustrated in Equation 6.10, where w_t is the weight vector produced in iteration $t-1$, w_{t+1} the resulting incremented weight vector of iteration t, and τ_t a loss function over a single feature vector x_t, its reference label y_t and a predicted label y'_t.

$$w_{t+1} = w_t + \tau_t y_t x_t \tag{6.10}$$

The algorithm's name is a reference to its behavior given the result of an iteration: if a loss is 0 at a given iteration t, then $w_{t+1} = w_t$ and its behavior is deemed *passive*, otherwise, it performs an *aggressive* change to vector w_{t+1} so that it ensures that the loss is 0. Unlike Stochastic Gradient Descent, the behavior of the Passive Aggressive algorithm is already said to prevent overfitting, and hence the algorithm does not require a regularisation term. Due to the nature of its weight vector update function, it also supports only Hinge and Squared Hinge losses.

6.4.2 Non-Linear Models

For non-linear Support Vector Machines, we have chosen two of the most widely used kernels in literature:

- **Polynomial**: Expands the feature space of a problem by applying a polynomial function of any degree to a pair of feature vectors. Equation 6.11 illustrates the generalised form of a polynomial kernel of degree d and trade-off coefficient c being estimated over two feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$.

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d \tag{6.11}$$

  In Equation 6.11, if $c = 0$, the kernel is called homogenous, and if $c \neq 0$, it is called heterogeneous. The parameter c determines a trade-off between higher and lower order terms in the polynomial components. For our experiments, we use a quadratic polynomial function i.e. $d = 2$.

- **Radial Basis Function**: Expands the feature space of a problem by applying a radial function over a distance measure between a pair of feature vectors. Equation 6.12 illustrates the generalised form of a Gaussian Radial Basis Function over to the Euclidean distance between two feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, which is used in our experiments.

$$K(\mathbf{x}_1, \mathbf{x}_2) = e^{\left(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)} \tag{6.12}$$

  In Equation 6.12, the γ coefficient determines the degree of influence that a given training example has in the learning process.

For linear models, we consider all possible combinations of supported loss functions, regularisation terms and learning algorithms in our evaluation. For non-linear models, we

consider the Polynomial and Radial Basis Function kernels. We evaluate the performance of our approach when trained over all 26 features described in Section 6.3.3, and use TRank-at-1:3 for evaluation. For the optimisation of hyper-parameters, including the maximum ranking position p, we use 10-fold cross-validation. To train our models, we use the scikit-Learn toolkit (Pedregosa et al., 2011).

The results obtained for the Stochastic Gradient Descent and Passive Agressive learning algorithms are presented in Tables 6.1 and 6.2, respectively. In Table 6.3 are presented the results obtained for the Support Vector Machines trained with the RBF and Polynomial kernels.

	L**1**			L**2**			Elastic Net		
Loss Func.	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
Hinge	**0.588**	**0.818**	0.920	0.583	0.817	0.919	0.583	0.817	0.919
Sq. Hinge	0.550	0.789	0.905	0.551	0.791	0.908	0.551	0.791	0.908
Mod. Huber	0.582	0.816	**0.923**	0.585	0.818	0.920	0.585	0.818	0.920

Table 6.1 TRank-at-n values for Stochastic Gradient Descent learning for all 26 features

Loss Func.	n=1	n=2	n=3
Hinge	0.576	0.811	0.917
Sq. Hinge	**0.587**	**0.819**	**0.919**

Table 6.2 TRank-at-n values for Passive Agressive learning for all 26 features

Kernel	n=1	n=2	n=3
RBF	**0.584**	**0.817**	**0.915**
Polynomial	0.538	0.778	0.888

Table 6.3 TRank-at-n values for non-linear kernel functions for all 26 features

Overall, the Squared Hinge loss pairs best with Passive Aggressive learning, while the Hinge loss pairs best with Stochastic Gradient Descent. Regularisation terms do not seem to greatly affect the performance of models trained with Stochastic Gradient Descent. For SVMs, the RBF kernel is much more suitable than the Polynomial kernel in this scenario. Nonetheless, the results reveal an absence of a statistically significant difference ($p > 0.05$) between the best performing settings of each learning algorithm.

We can conclude that both linear and non-linear models are suitable for Boundary Ranking. For the subsequent experiments, we train linear Boundary Ranking models using Stochastic Gradient Descent, Hinge loss function and L1 regularisation.

6.5 Feature Selection Assessment

We also assess the performance of several feature configurations with our approach. Selecting the appropriate features for a problem can greatly influence a model's performance. We consider 29 sets of features:

1. **All Features**: The set of all 26 features described in Section 6.3.3.

2. **Collocational Features**: A set of nine features that estimate the language model probabilities $P\left(S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}\right)$ of all n-grams with left (l) and right (r) window sizes less or equal to two. As shown in the experiments of Jauhar and Specia (2012) and Glavaš and Štajner (2015), such collocational features can be used to produce state-of-the-art results for Substitution Ranking.

3. **CW Features**: The set of six features used in the experiments that compare several Complex Word Identification approaches of Shardlow (2013a). They are: candidate frequency with respect to the SUBTLEX corpus, the number of subtitles in the the SUBTLEX corpus in which a candidate appears, word length, number of syllables, and number of senses and synonyms as registered in WordNet.

4. **Horn Features**: A set of 15 features similar to the ones used in the Substitution Ranking experiments of Horn et al. (2014). They are: the candidate's alignment probability with the target complex word, as estimated by an IBM model trained over the parallel corpus of Wikipedia and Simple Wikipedia sentences, the candidate's frequency in Wikipedia and Simple Wikipedia, the language model probability of the target sentence with its target complex word replaced by the candidate, as estimated by language models trained over Wikipedia, Simple Wikipedia and the corpus described in Section 6.3.3, and the frequencies $f\left(S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}\right)$ of all n-grams with left and right window sizes less or equal to two in Simple Wikipedia.

5. **Best n Features**: Using a feature selection technique, we select the best $1 \leq n \leq 25$ features from the total set of features described in Section 6.3.3.

In order to select the best n features from the original set of 26, we use univariate feature selection through the ANOVA F-Test (Lomax, 2007). The F-Test was initially conceived as a approach for the problem of deciding if there is a statistically significant difference between the means of various populations, and it makes such a decision by evaluating the behavior of variances *between* samples of the populations with the variances *within* them. It draws an F

measure over the population samples, which is calculated as shown in Equation 6.13, and then compares it to a critical F value determined over the distribution of sample values.

$$F = \frac{\left(\frac{SSTR}{c-1}\right)}{\left(\frac{SSE}{N-c}\right)} \tag{6.13}$$

In Equation 6.13, c is the number of samples for each population, N the total number of samples, $SSTR$ the variation in the data samples between different populations, and SSE the sum of the variation within the data samples of each population. $SSTR$ and SSE can be calculated as shown in Equations 6.14 and 6.15, where X_{ij} is the value of the ith sample of population j, r_j is number of samples in population j, $\overline{X}_j$ is the average of the samples in population j, and $\overline{\overline{X}}$ is the average of all samples.

$$SSTR = \sum r_j \left(\overline{X}_j - \overline{\overline{X}}\right)^2 \tag{6.14}$$

$$SSE = \sum \sum \left(X_{ij} - \overline{X}_j\right)^2 \tag{6.15}$$

In statistics, the estimated F and the critical F values are compared: if the estimated F is greater than the critical F, then the hypothesis that all (or at least two) populations have equal means is discarded, otherwise, it is confirmed. In feature selection, however, we discard the need for a critical F value, and instead quantify the F statistic as a metric that represents the capability of a given feature to separate two or more classes in a classification problem. In our binary classification setup for SR, the F statistic represents the capability of a given feature to discern between candidates that are simple enough to simplify a target complex word, and those that are not.

The TRank-at-1 : 3 scores obtained with each feature set are presented in Tables 6.4 and 6.5. The results show that feature selection can greatly improve on the performance of our approach, but selecting an exceedingly small or large set of features can actually compromise it. We have also found that collocational features extracted from SubIMDB are very effective for SR, and can outperform almost all other sets of features. Such phenomenon is in accordance with the findings of the user study in Section 3.3, where it is suggested that word simplicity is context-dependent.

6.6 Performance Comparison

Here we compare the performance of Boundary Ranking with other ranking strategies. In this experiment, we trained a linear model ranker with the same collocational features described

Feature Set	TRank-at-1	TRank-at-2	TRank-at-3
All	0.529	0.774	0.891
Colloc.	**0.655**	**0.863**	**0.940**
CW	0.494	0.742	0.880
Horn	0.635	0.854	0.932

Table 6.4 TRank-at-*n* values obtained with different feature sets

Features	TRank-at-1	TRank-at-2	TRank-at-3
1	0.622	0.849	0.935
2	0.646	0.862	0.939
3	0.647	0.862	**0.942**
4	0.646	0.860	0.940
5	0.647	0.861	0.940
6	0.649	0.862	0.940
7	0.649	**0.863**	0.941
8	0.647	0.862	0.941
9	0.650	0.859	0.935
10	0.646	0.853	0.933
11	0.650	**0.863**	0.941
12	**0.655**	0.860	0.937
13	**0.655**	0.860	0.937
14	0.647	0.858	0.937
15	0.652	0.861	0.938
16	0.652	0.860	0.938
17	0.650	0.860	0.937
18	0.653	0.861	0.940
19	**0.655**	0.861	0.939
20	0.654	0.861	0.939
21	0.652	0.857	0.935
22	0.589	0.816	0.904
23	0.595	0.825	0.913
24	0.573	0.811	0.920
25	0.561	0.795	0.910

Table 6.5 TRank-at-*n* values obtained with different numbers of selected features

in Section 6.3.3. Since the collocational features require a language model trained over some type of corpus, we selected three of the most widely used corpora in LS to train them with:

- **Simple Wikipedia**: Composed of 9,132,292 words taken from Simple Wikipedia articles.
- **SUBTLEX**: Composed of 62,498,097 words taken from assorted subtitles.
- **SubIMDB**: Composed of over 228,622,257 words extracted from movies and series for family and childrem.

By using different corpora to extract features from, we can assess the quality of language model probabilities from the SubIMDB corpus. Using these three sets of features, we train three variants of not only our Boundary Ranking model, but also the Glavas and the SVMRank rankers. This way, we can compare the performance of these approaches under the same settings.

We include the SemEval 2012 systems UOW-SHEF and Google 1T described in Section 6.3.4 in our performance comparison also. The TRank-at-1, 2 and 3 scores obtained are presented in Table 6.6. Notice that the authors of the proceedings of SemEval 2012 do not include values for TRank-at-2 and 3 for the systems submitted.

Approach	TRank-at-1	TRank-at-2	TRank-at-3
Google1T	0.585	-	-
Best Semeval	0.602	-	-
Glavas (Simple Wiki)	0.587	0.814	0.919
Glavas (SUBTLEX)	0.609	0.820	0.913
Glavas (SubIMDB)	0.591	0.826	0.922
SVMRank (Simple Wiki)	0.599	0.820	0.927
SVMRank (SUBTLEX)	0.628	0.841	0.926
SVMRank (SubIMDB)	0.644	0.853	0.933
Boundary (Simple Wiki)	0.610	0.825	0.932
Boundary (SUBTLEX)	0.637	0.857	0.935
Boundary (SubIMDB)	**0.654**	**0.858**	**0.936**

Table 6.6 TRank-at-n values obtained in the performance comparison between Boundary Ranking models and baselines

The results show that our Boundary Ranking approach outperforms not only the best performing systems of SemEval 2012, but also both Glavas and SVMRank rankers in all feature configurations. F-tests reveal that the differences between all rankers is statistically significant ($p < 0.05$).

We can also notice from the results that, when trained with features extracted from the SubIMDB corpus, both SVMRank and Boundary Ranking performs considerably better than when trained with the Simple Wikipedia and SUBTLEX corpora. This suggests that SubIMDB is a more reliable resource for the training of Lexical Simplification systems that aim to address the needs of non-native English speakers.

6.7 Conclusions

In this Chapter, we have presented a novel supervised approach to the task of Substitution Ranking. In Section 6.1 we discussed some of the misconceptions under which even modern SR approaches are conceived and evaluated.

In an effort to address their limitations, we then proposed Boundary Ranking: a flexible supervised strategy for the training of ranking models that exploits a binary classification setup inferred from ranking examples found in most LS datasets available. A Boundary Ranker attempts to produce a hyperplane over a feature space that separates candidate substitutions that are simple enough to replace a complex target word from those that are not. It then uses the projected distance between the feature values that describe an unseen candidate and the hyperplane to determine how simple it is, which in turn allows us to rank candidates according to their simplicity.

In order to evaluate our new strategy, we have conducted experiments which measure its performance in various settings. We found that using distinct loss functions, regularisation terms, learning algorithms and kernels can greatly influence the performance of our approach, and that maximising those parameters during training plays a very important role in ensuring that it performs well. We have also evaluated the performance of various feature combinations, which revealed the potential of collocational features in estimating simplicity for non-native English speakers. By comparing the performance of collocational features extracted from different corpora, we reveal that SubIMDB, the corpus of subtitles described in Section 8.1, is able to capture simplicity more effectively than all other corpora, given the needs of non-native English speakers. Combining hyper-parameter optimisation and collocational features estimated over SubIMDB, we were able to train a Boundary Ranker that outperforms the most effective supervised and unsupervised rankers in literature.

Chapter 7

Ranking Models for Substitution Selection

Substitution Selection (SS) is the task of, given a set of n substitution candidates $\{c_1, ..., c_n\}$ for a complex word t in sentence S, selecting which candidates c_i can replace t while ensuring that both the grammaticality and meaning of S will be preserved. The final goal of the task is to prevent an LS approach from incorrectly replacing ambiguous complex words that have two or more meanings or pertain to two or more grammatical classes. Consider, for example, that a Complex Word Identification system judged that the word "*yield*" poses a challenge, and the candidate substitutions produced by a Substitution Generation approach are "*proceeds*", "*production*", "*payoff*", "*concede*", "*succumb*" and "*grant*". Now suppose that the three sentences in Table 7.1, which portray the use of "*yield*" in three very distinct contexts, must be simplified with the substitutions produced.

Sentence	Form	Meaning
The king will **yield** you a fortress.	verb	To forfeit something to a third party.
The average **yield** was about 10%.	noun	Income or profit from a transaction or sale.
He may **yield** to the pain soon.	verb	To be fatally overwhelmed by something.

Table 7.1 Sentences containing the word "*yield*"

One will notice that all three senses of "*yield*" are very distinct, and that some of the candidate substitutions available do not share the same meaning as the complex word in each context. The examples in Table 7.2 illustrate the results that would be expected from an arguably ideal SS approach.

The examples in Table 7.2 highlight the importance of Substitution Selection: if a substitution were to be selected at random for each of the aforementioned sentences, the chance of "*yield*" being incorrectly replaced would be very high.

Sentence	Selected Candidates
The king will **yield** you a fortress.	concede, grant
The average **yield** was about 10%.	proceeds, production, payoff
He may **yield** to the pain soon.	succumb

Table 7.2 Selected substitutions for "*yield*"

Considering this example, perhaps the most intuitive approach to the ambiguity problem in LS would be to employ a state-of-the-art Word Sense Disambiguation approach during SS. But as discussed in the survey of Chapter 2, although the precision of WSD systems have been consistently increasing throughout the years Navigli (2009), even the most effective approaches for the task are not reliable enough to be used in SS.

Yet the biggest downside inherent to the use of WSD systems in LS lies not the unsatisfactory performance of such systems, but rather in the fact that they address the problem of SS as a sense label classification task. WSD systems resort to large linguistic databases to determine the sense of a given word, a strategy that works under the premises that such databases have high enough coverage of senses, and that they register each and every word of a language's vocabulary that can possibly be assigned to such senses.

In order to address the aforementioned problems, we propose a new setup for the task of SS. Instead of deciding which words can be considered synonyms of a target word in a sentence, we attempt to rank substitution candidates according to their likelihood of fitting the context in which the target word is found. Once the candidates are ranked, the selector can then discard the ones which are less likely to fit.

Our ranking setup has several advantages over using WSD systems:

- It allows for both supervised and unsupervised ranking techniques to be used. Some examples of techniques that can be used are Machine Learning-based and metric-based approaches.

- It allows for ranking techniques to exploit several types of features that represent the relation between the substitution candidates and the target word's context. N-gram frequency counts, semantic similarity metrics and translation probability scores are some examples.

- It allows for one to easily customise the behavior of the SS approach. Since each and every candidate substitution will be have a certain likelihood of fitting a given context, the user can create either an optimistic selector that keeps only the highest ranked candidates, or a conservative selector which discards only the lowest ranked candidates.

As shown by the results discussed in Chapter 6, unsupervised metric-based ranking strategies can offer competitive performance, but cannot outperform more sophisticated supervised methods. Supervised ranking approaches, do, however, require annotated data for models to be trained. This type of data is scarce in the domain of LS, and can be expensive to produce.

In this Chapter, we introduce a novel, unsupervised approach for SS based on the supervised Boundary Ranking strategy described in Chapter 6. We henceforth refer to Substitution Selection systems that take the task as a ranking problem as "ranking-based selectors".

7.1 Unsupervised Boundary Ranking

Ranking substitution candidates can be done in multiple ways, both supervised and unsupervised. As discussed in Chapter 2, in LS, unsupervised ranking approaches take the form of simple metric-based rankers. Such rankers often order candidates according to a manually crafted metric that combines a few features, such as n-gram frequencies in a corpus. Although the results reported in the English Lexical Simplification task of SemEval 2012 show that metric-based strategies can be quite effective in Substitution Ranking, they do not allow for one to automatically learn patterns from unannotated data. In this Section, we introduce the first unsupervised SS strategy capable of doing so. Before explaining how we create an unsupervised ranking-based selector, we elaborate on how we could use the supervised Boundary Ranking approach described in Section 6.2 in SS.

In Boundary Ranking, substitution candidates are ranked according to their distance from a hyperplane (or "boundary") learned over annotated ranking data. In order to learn the hyperplane, a Boundary Ranker first infers a set of binary classification training instances from the annotated data based on an optimisable parameter p, which determines the maximum ranking position that receives label 1, and a function $\mathbf{x} = f(\mathrm{c}_i)$, which represents a set of feature values that describe a candidate c_i with respect to target complex word t and sentence S.

As discussed in Section 6.2, Boundary Ranking could be used in any ranking task: one would only need to use the appropriate dataset and features for training. If we take SS to be a ranking problem, we can consequently use Boundary Ranking to train a supervised ranking-based selector over features that capture grammaticality and meaning preservation. Consider, for example, the candidate substitutions in Table 7.3 for target word "*perched*" in sentence "*The cat perched on the window*", ranked by how well they fit the context of the target with respect to grammaticality and meaning preservation.

Candidate	Rank
sat	1
rested	2
roost	3
perch	4

Table 7.3 Candidate substitutions ranked by how well they fit the context of "*perched*"

For this training instance, if the parameter p of a Boundary Ranker is set to 2, then "*sat*" and "*rested*" would receive label 1, while "*roost*" and "*perch*" would receive label 0. Once the binary classification training instances are produced, the hyperplane that separates candidates that are and are not good enough to replace the target word can be estimated through the use of any learning technique capable of doing so.

This example, however, only illustrates how a Boundary Ranker could be trained in a supervised fashion for SS, not unsupervised. In a supervised scenario, a Boundary Ranker for SS would require a set of several training instances with candidate substitutions ranked by human annotators by how likely they are of fitting the context of a target word. Without annotated data, a Boundary Ranker can only be trained if we explore the Robbins-Sturgeon hypothesis, which we propose.

In his book Jitterbug Perfume (Robbins, 2003), Tom Robbins presents his opinion on semantic relations between words by stating that "*There are no such things as synonyms! He practically shouted. Deluge is not the same as flood*". A similar statement was made by Theodore Sturgeon, author of several science fiction and horror stories such as Not Without Sorcery (Sturgeon, 1948). In an interview conducted by David D. Duncan, Sturgeon said "*Here's the point to be made - there are no synonyms. There are no two words that mean exactly the same thing. I don't care about the dictionaries of synonyms and antonyms. If there were two words that meant exactly the same thing, there wouldn't be two words. That means that every word you use has a certain amount of semantic or psychological freight that it carries that makes it different from other words*".

As a summary of the authors' quotes, we introduce the Robbins-Sturgeon hypothesis. It states that a word is irreplaceable in the context it was originally presented, implying that no other word in a vocabulary can correctly convey the same information. Although the progress made by modern LS approaches such as the ones of Horn et al. (2014) and Glavaš and Štajner (2015) show that exploring synonymy relations between words is very useful in making texts easier to read, it is still worth exploring the Robbins-Sturgeon hypothesis in unsupervised Boundary Ranking for SS.

If we take the Robbins-Sturgeon hypothesis to be correct, we can assume that a given target complex word is the only word suitable to replace itself. In the binary classification setup of a Boundary Ranker, this would mean that the only candidate substitution which would receive label 1 would be target word itself, while any other candidates, regardless of how well one may judge them to be able to replace the target word, would receive label 0. With these settings, we would not require any annotated data: the substitution candidates could be produced by an unsupervised approach. Notice that, in this case, the p value is equal to one, and hence cannot be optimised like in a supervised setting.

Consider, for example, that the words "*produce*", "*give*" and "*destabilise*" have been generated as candidate substitutions for the target word "*yield*" in a certain sentence. Using this information, we can produce four binary classification training instances, as illustrated in Table 7.4.

By repeating this process for other complex words, one should produce enough training instances for an unsupervised Boundary Ranker to be trained in reliable fashion. Once the model is trained, one can then rank the candidates produced for an unseen simplification problem, then finally select a proportion of them to be carried on to Substitution Ranking.

In order to evaluate the performance of our strategy, we have conducted several experiments. We test the Robbins-Sturgeon hypothesis in practice (Section 7.3), benchmark various ranking-based selectors (Section 7.4), investigate how the proportion of selected candidates influence the performance of our approach (Section 7.5), and compare our approach to current state-of-the-art SS strategies in a full pipeline evaluation that pairs selectors with various SG and SR strategies (Section 7.6).

Candidate	Label
yield	1
give	0
produce	0
destabilise	0

Table 7.4 Binary classification training instances for unsupervised Boundary Ranking

7.2 Experimental Settings

7.2.1 Approaches

In our experiments, we compare the performance of several ranking approaches to SS, as well as various other selectors from previous work. The ranking approaches included in our experiments are:

- **Boundary Ranking**: Employs Boundary Ranking. It can be trained in both supervised and unsupervised fashion. For supervised SS Boundary Rankers, the rankings in datasets such as LexMTurk (Horn et al., 2014) or LSeval (De Belder and Moens, 2012b) could be used. Given the results reported in Chapter 6, we choose a linear model trained with Stochastic Gradient Descent, Hinge loss function and L1 regularisation for both supervised and unsupervised settings. We optimise all hyper-parameters with 10-fold cross validation.

- **Support Vector Machines**: Employs the supervised ranking strategy proposed by Joachims (2002), which uses Support Vector Machines in a setup that minimises a loss function with respect to a ranking model. This strategy is the one used in the Substitution Ranking approach of Horn et al. (2014). We optimise all hyper-parameters with 10-fold cross validation.

- **Metric-Based**: Ranks candidate substitutions according to a certain metric that represents their likelihood of fitting in the same context of a target word. We use as metrics each and every feature described in Section 7.2.2.

The SS systems from previous work that we include in our performance comparisons are:

- **First Sense (First)**: Selects only those words which are listed as synonyms under the first WordNet sense of the target word.

- **Random Sense (Random)**: Selects only those words which are listed as synonyms under a random WordNet sense of the target word.

- **Lesk Algorithm (Lesk)**: Uses the algorithm introduced by Lesk (1986), described in Section 5.4.

- **Path Similarity (Path)**: Uses the algorithm introduced by Leacock and Chodorow (1998), described in Section 5.4.

- **Word Clustering (Belder)**: Uses a strategy similar to the one presented by De Belder and Moens (2010), described in Section 5.4.

- **Co-Occurrence Model Filtering (Biran)**: Uses the strategy introduced by Biran et al. (2011), described in Section 5.4.

Since SS is performed after candidates are generated, we evaluate the SS approaches above when paired with four SG approaches:

- **Paetzold**: Employs the SG strategy described in Chapter 5, which extracts candidate substitutions from retrofitted context-aware embedding models. We use a model trained with the word2vec toolkit. The corpus used contains around 7 billion words, and is composed of texts extracted from the UMBC webbase, News Crawl, the SUBTLEX and SubIMDB corpora, as well as Wikipedia and Simple Wikipedia. The POS tags required for the model were produced through the use of the Stanford Parser. We use the CBOW architecture and $1,300$ dimensions for the embeddings vector.

- **Devlin** (Devlin and Tait, 1998): Generates candidate substitutions by extracting synonyms from WordNet, as described in Section 5.4.

- **Kauchak** (Horn et al., 2014): Generates candidate substitutions from complex-to-simple parallel corpora, as described in Section 5.4.

- **All**: Combines the substitutions generated by all generators above.

7.2.2 Features

We use seven features for the training of all supervised ranking-based selectors:

- **N-gram Probabilities**: Language model log-probabilities of the following eight n-grams: $S_{i-1}c_i$, c_iS_{i+1}, $S_{i-1}c_iS_{i+1}$, $S_{i-2}S_{i-1}c_i$, $S_{i-2}S_{i-1}c_iS_{i+1}$, $c_iS_{i+1}S_{i+2}$, $S_{i-1}c_iS_{i+1}S_{i+2}$ and $S_{i-2}S_{i-1}c_iS_{i+1}S_{i+2}$ where c is a candidate substitution, and i the position of the target complex word in sentence S. We use a 5-gram language model trained over SubIMDB corpus, described in Section 8.1, with the SRILM toolkit.

- **Semantic Similarity**: The word embeddings cosine similarity between the target complex word and a candidate. We use the same retrofitted context-aware model used by the Paetzold generator.

- **Part-of-Speech Conditional Probability**: The conditional probability of the candidate substitution receiving the same POS tag as the target word. To calculate this feature, we learn the word-to-tag probability distribution $P(c|p_t)$, described in Equation 7.1, of all words in the corpus used to train our retrofitted context-aware word embeddings model.

$$P(c|t_i) = \frac{C(c,t_i)}{\sum_{t \in T} C(c,t)} \tag{7.1}$$

  In Equation 7.1, c is a candidate substitution, t_i is the POS tag of the target word in a given instance, $C(c,t)$ the number of times c received tag t in the training corpus, and T the set of all existing POS tags.

We choose these features because they have been frequently used in the creation of some of the most effective LS systems to date.

7.2.3 Datasets

To evaluate the systems, we use the LexMTurk corpus, which is composed of 500 instances. Each instance is composed of a sentence extracted from Wikipedia, a target complex word, and 50 simpler alternatives suggested by turkers from the Amazon Mechanical Turk. We take such suggestions as our gold-standard.

7.2.4 Evaluation Metrics

For the task of Substitution Selection alone, we use the same four distinct evaluation metrics introduced in Chapter 5:

- **Potential**: The proportion of instances in which at least one of the substitutions generated is present in the gold-standard.
- **Precision**: The proportion of generated substitutions that are present in the gold-standard.
- **Recall**: The proportion of gold-standard substitutions that are included in the generated substitutions.
- **F1**: The harmonic mean between Precision and Recall.

For the full pipeline evaluation, in which various combination of SG, SS and SR approaches are compared, we use the following three evaluation metrics:

- **Precision**: The proportion of instances in which the target word was replaced with any of the candidates in the dataset, including the target word itself.

- **Accuracy**: The proportion of instances in which the target word was replaced with any of the candidates in the dataset, except for the target word itself.

- **Changed Proportion**: The proportion of times in which the target word was replaced with a different word.

7.3 Testing the Robbins-Sturgeon Hypothesis

In our first experiment, we evaluate how well the Robbins-Sturgeon hypothesis works in practice. We evaluate the performance of our unsupervised Boundary Ranker in selecting substitutions generated by each and every SG approach described in Section 7.2.1. To train our selector, we perform the following three steps:

1. For each instance of the LexMTurk dataset, create a positive binary classification training instance by assigning label 1 to the target word itself.

2. For each instance of the LexMTurk dataset, run all SG approaches and create negative binary classification instances by assigning label 0 to all candidate substitutions produced that are different from the target word itself.

3. Train the binary classifier over the instances produced.

We also explored using the Robbins-Sturgeon hypothesis in supervised ranking for SS. To do so, we created two distinct versions of the LexMTurk dataset:

- **Best-First**: The candidate substitutions in the gold-standard for each instance are ranked according to the number of times they have been suggested by human annotators. The candidate suggested most frequently is ranked first, and is consequently deemed the one that best fits the context in which the target complex word is inserted.

- **Target-First**: Identical to the original version, except the target complex word of each instance is ranked first, and the ranking of the remaining candidates is increased by one. This version of the dataset exploits the Robbins-Sturgeon hypothesis, deeming the target word itself to be the its best candidate substitution.

We then split both LexMTurk versions in a training and a test set, composed both of 250 instances each. Finally, we use these datasets to train four supervised ranking-based selectors: two Boundary Rankers and two SVM rankers trained over the training portion of the Best-First and Target-First datasets.

All aforementioned supervised and unsupervised rankers are evaluated in the test set of the original (Best-First) version of LexMTurk. All rankers extract the same number of "gold counts" for each instance of the test set. The gold count of a given instance is the number of distinct candidate substitutions suggested by the annotators who created the LexMTurk corpus. Consider, for example, the instance in LexMTurk that refers to target word "*vital*" in sentence "*Photosynthesis is vital for life on Earth*". The candidates substitutions suggested by annotators are five: "*important*, "*critical*, "*essential*, "*necessary* and "*needed*. Consequently, the gold count for this instance is five.

This strategy has the goal of evaluating how these ranking-based selectors would perform in case the exact number of words that can replace a given target word is known. It is important to mention that, in a realistic scenario, these gold-counts are not available. We investigate how the proportion of candidates discarded influence the performance of our ranking-based selectors in Section 7.5.

The results obtained for each and every Substitution Generation approach are illustrated in Tables 7.5 to 7.8. The meaning of the identifiers in columns one and two are:

- **BF**: The ranker was trained over the Best-First version of LexMTurk, and hence did not exploit the Robbins-Sturgeon hypothesis.
- **TF**: The ranker was trained over the Target-First version of LexMTurk, and hence have exploited the Robbins-Sturgeon hypothesis.
- **U**: The ranker was trained in unsupervised fashion.
- **S**: The ranker was trained in supervised fashion.

For supervised rankers, using the target word itself as its best candidate substitution did not translate to any statistically significant ($p < 0.05$) improvements in performance compared to using the Best-First training set. For the majority of generators, training the supervised rankers over the Target-First version of LexMTurk made their F1 scores either equal or smaller than the ones obtained by rankers trained over the Best-First version.

In contrast, our unsupervised Boundary Ranker highlights the potential of the Robbins-Sturgeon hypothesis in SS. Without the need for manually annotated data, our approach was able to outperform all supervised rankers in almost all evaluation metrics for all generators.

It even outperforms the SVM rankers, which use a much more sophisticated modelling strategy than Boundary Ranking. These results are further evidence of the effectiveness of Boundary Ranking in learning ranking models, and suggest that unannotated bodies of text do carry valuable information about grammaticality and meaning preservation, which can be harnessed with the help of a suitable learning strategy.

Training	Dataset	Approach	Potential	Precision	Recall	F1
S	BF	Boundary	**0.608**	**0.194**	**0.085**	**0.118**
S	TF	Boundary	0.604	0.192	**0.085**	**0.118**
S	BF	SVM Rank	0.592	0.190	0.084	0.116
S	TF	SVM Rank	0.596	0.189	0.083	0.116
U	-	Boundary	**0.608**	**0.194**	**0.085**	**0.118**

Table 7.5 Performance results for the Devlin generator

Training	Dataset	Approach	Potential	Precision	Recall	F1
S	BF	Boundary	**0.820**	0.229	0.124	0.161
S	TF	Boundary	0.800	0.230	0.125	0.162
S	BF	SVM Rank	0.808	0.229	0.124	0.161
S	TF	SVM Rank	0.800	0.225	0.123	0.159
U	-	Boundary	0.812	**0.231**	**0.126**	**0.163**

Table 7.6 Performance results for the Kauchak generator

Training	Dataset	Approach	Potential	Precision	Recall	F1
S	BF	Boundary	0.764	**0.192**	**0.124**	**0.151**
S	TF	Boundary	0.760	0.190	0.123	0.149
S	BF	SVM Rank	0.748	0.188	0.122	0.148
S	TF	SVM Rank	0.748	0.188	0.122	0.148
U	-	Boundary	**0.768**	**0.192**	**0.124**	**0.151**

Table 7.7 Performance results for the Paetzold generator

7.4 Benchmarking Ranking Strategies

Now that we have evidence that our unsupervised approach offers competitive performance for SS, we assess how well it performs in comparison to several other supervised and unsupervised ranking strategies for the task. For evaluation, we use the same settings used in our last experiment: each ranker selects the number of "gold counts" of each instance in the

Training	Dataset	Approach	Potential	Precision	Recall	F1
S	BF	Boundary	0.928	0.217	0.210	0.213
S	TF	Boundary	0.940	0.224	0.217	0.220
S	BF	SVM Rank	0.920	0.229	0.223	0.226
S	TF	SVM Rank	0.880	0.212	0.206	0.209
U	-	Boundary	**0.944**	**0.247**	**0.240**	**0.244**

Table 7.8 Performance results for all generators combined

test portion of the Best-First version of LexMTurk. In our second experiment, we compare the performance of 13 distinct rankers:

- **Unsupervised Boundary Ranker**: The approach introduced in this Chapter, trained in same way as in the previous experiment.
- **Supervised Boundary Ranker**: The same approach described in the previous experiment, trained over the training portion of the Best-First version of LexMTurk.
- **Supervised SVM Ranker**: The same approach described in the previous experiment, trained over the training portion of the Best-First version of LexMTurk.
- **Metric-Based Rankers**: A total of 10 unsupervised rankers, where each ranker uses as a metric one of the features described in Section 7.2.2. The features are eight n-gram frequencies under a (l,r) configuration, in which l and r are the number of tokens to the left and right, respectively, the word vector similarity between target and candidate, and target POS tag conditional probability.

We evaluate the performance of the ranking-based selectors over the candidate substitutions produced by each and every generator described in Section 7.2.1. The results are presented in Tables 7.9 through 7.12.

It can be observed from the results that n-gram frequency metric-based rankers can be effective in SS, since, although simple in nature, they still offer very competitive results in comparison to the supervised rankers. Nonetheless, none of the metric-based approaches were able to outperform our unsupervised approach, which still offers the highest F1 scores for all generators evaluated.

7.5 Selection Proportion Assessment

While the previous experiments revealed the potential of ranking-based selectors, they did not include any comparisons between ranking models and other SS approaches. Different

Training	Approach	Potential	Precision	Recall	F1
U	Boundary	0.608	**0.194**	0.085	**0.118**
S	Boundary	0.608	**0.194**	0.085	**0.118**
S	SVM Rank	0.592	0.190	0.084	0.116
U	N-Gram (0, 1)	**0.628**	0.178	**0.088**	**0.118**
U	N-Gram (0, 2)	0.624	0.177	0.088	0.117
U	N-Gram (1, 0)	0.620	0.175	0.087	0.116
U	N-Gram (1, 1)	0.624	0.177	0.088	0.117
U	N-Gram (1, 2)	0.620	0.176	0.087	0.117
U	N-Gram (2, 0)	0.620	0.175	0.087	0.116
U	N-Gram (2, 1)	0.624	0.177	0.088	0.117
U	N-Gram (2, 2)	0.620	0.176	0.087	0.117
U	Embed. Sim.	0.608	0.169	0.084	0.112
U	POS Prob.	0.596	0.168	0.083	0.112

Table 7.9 Performance results for the Devlin generator

Training	Approach	Potential	Precision	Recall	F1
U	Boundary	0.812	**0.231**	**0.126**	**0.163**
S	Boundary	**0.820**	0.229	0.124	0.161
S	SVM Rank	0.808	0.229	0.124	0.161
U	N-Gram (0, 1)	**0.820**	0.205	0.120	0.152
U	N-Gram (0, 2)	**0.820**	0.206	0.121	0.153
U	N-Gram (1, 0)	0.812	0.202	0.119	0.150
U	N-Gram (1, 1)	**0.820**	0.209	0.123	0.155
U	N-Gram (1, 2)	**0.820**	0.209	0.123	0.155
U	N-Gram (2, 0)	0.812	0.202	0.119	0.150
U	N-Gram (2, 1)	**0.820**	0.209	0.123	0.155
U	N-Gram (2, 2)	**0.820**	0.209	0.123	0.155
U	Embed. Sim.	0.808	0.208	0.123	0.154
U	POS Prob.	0.772	0.195	0.115	0.145

Table 7.10 Performance results for the Kauchak generator

Training	Approach	Potential	Precision	Recall	F1
U	Boundary	0.768	**0.192**	**0.124**	**0.151**
S	Boundary	0.764	**0.192**	**0.124**	**0.151**
S	SVM Rank	0.748	0.188	0.122	0.148
U	N-Gram (0, 1)	0.768	0.178	0.123	0.145
U	N-Gram (0, 2)	0.768	0.178	0.123	0.146
U	N-Gram (1, 0)	0.776	0.178	0.123	0.145
U	N-Gram (1, 1)	0.772	0.178	0.123	0.146
U	N-Gram (1, 2)	0.772	0.178	0.123	0.145
U	N-Gram (2, 0)	**0.776**	0.178	0.123	0.145
U	N-Gram (2, 1)	0.772	0.178	0.123	0.146
U	N-Gram (2, 2)	0.772	0.178	0.123	0.146
U	Embed. Sim.	0.748	0.171	0.119	0.140
U	POS Prob.	0.728	0.168	0.117	0.138

Table 7.11 Performance results for the Paetzold generator

Training	Approach	Potential	Precision	Recall	F1
U	Boundary	**0.944**	**0.247**	**0.240**	**0.244**
S	Boundary	0.928	0.217	0.210	0.213
S	SVM Rank	0.920	0.229	0.223	0.226
U	N-Gram (0, 1)	0.816	0.169	0.165	0.167
U	N-Gram (0, 2)	0.828	0.171	0.167	0.169
U	N-Gram (1, 0)	0.868	0.176	0.172	0.174
U	N-Gram (1, 1)	0.892	0.198	0.193	0.195
U	N-Gram (1, 2)	0.884	0.199	0.195	0.197
U	N-Gram (2, 0)	0.868	0.177	0.172	0.174
U	N-Gram (2, 1)	0.896	0.199	0.195	0.197
U	N-Gram (2, 2)	0.884	0.201	0.196	0.199
U	Embed. Sim.	0.816	0.176	0.172	0.174
U	POS Prob.	0.748	0.160	0.156	0.158

Table 7.12 Performance results for all generators combined

from rankers, other selectors perform a binary decision over each candidate substitution generated: can it replace the target word without compromising the sentence's grammaticality nor changing its meaning, or not? Consequently, for a ranker and a classifier to be fairly compared, the ranker would have to select a predetermined portion of candidate substitutions.

In our third experiment, we assess how the performance of a ranking selector is affected by the proportion of candidate substitutions selected for each instance of a dataset based on the gold-standard. The gold-standard used is the test portion of LexMTurk. We assess the performance of four rankers:

- **Unsupervised Boundary Ranker**: The approach introduced in this Chapter, trained in same way as in the experiment of Section 7.3.
- **Supervised Boundary Ranker**: The same approach described in the experiment of Section 7.3, trained over the training portion of the Best-First version of LexMTurk.
- **Supervised SVM Ranker**: The same approach described the experiment of Section 7.3, trained over the training portion of the Best-First version of LexMTurk.
- **Metric-Based Ranker**: The overall best performing metric ranker from the experiment of Section 7.3. It ranks candidates according to the n-grams composed of two tokens to the right and the left of the target word.

We test selection proportions that range from 10% to 90% of the candidate substitutions generated, in intervals of 10%. The same proportion of candidates is selected for all instances in the test set. The Potential, Precision, Recall and F1 scores obtained for each generator are illustrated in Figures 7.1 through 7.4.

The score graphs reveal that, in most cases, the highest performing selector remains so independently of the proportion of substitutions selected. This phenomenon grants consistency to the previous two experiments, further highlighting the potential of our unsupervised approach.

It can also be observed that the proportion that yields the highest F1 scores is not consistent throughout the generators. If 80% of the candidates are selected, for example, the selectors would perform very well for the Devlin generator, but very poorly for all generators combined. In contrast, the opposite would happen if only 30% of candidates were selected. We hypothesise that this phenomenon is linked to the average number of candidates produced by each approach. While the Devlin generator, which takes synonyms from WordNet, produces an average of 10 candidates per complex word, combining all generators yields an average of 30. Since WordNet is a manually created lexicon, the Devlin

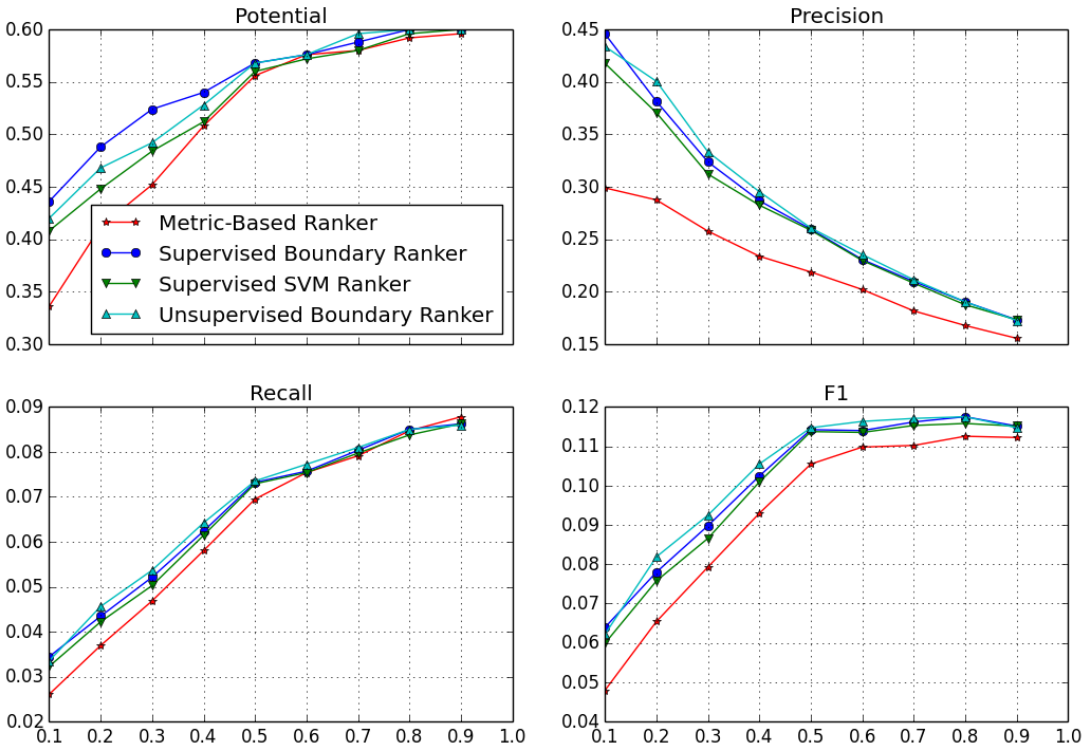

Fig. 7.1 Percentage selection performance results for the Devlin generator

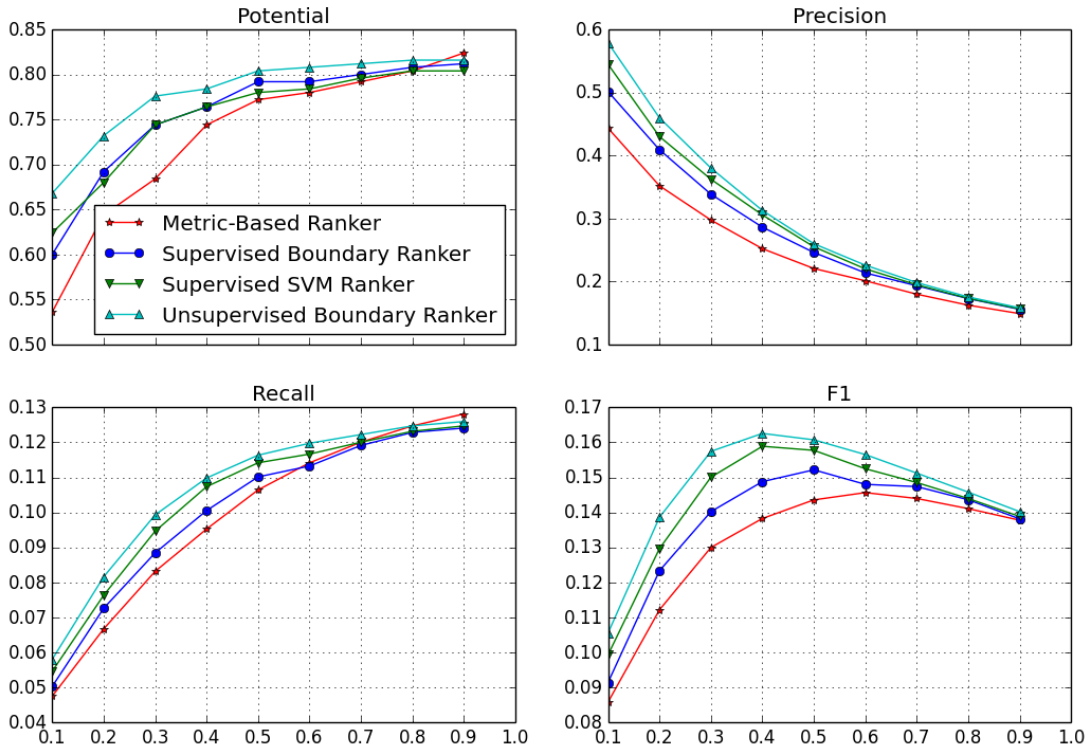

Fig. 7.2 Percentage selection performance results for the Kauchak generator

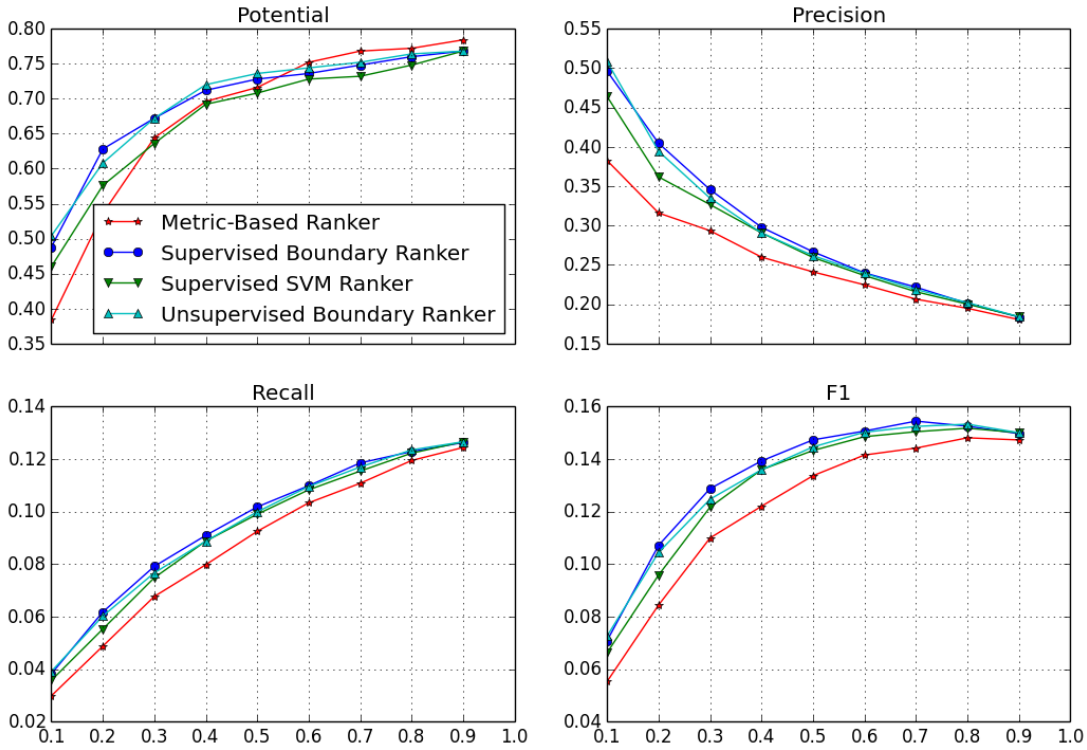

Fig. 7.3 Percentage selection performance results for the Paetzold generator

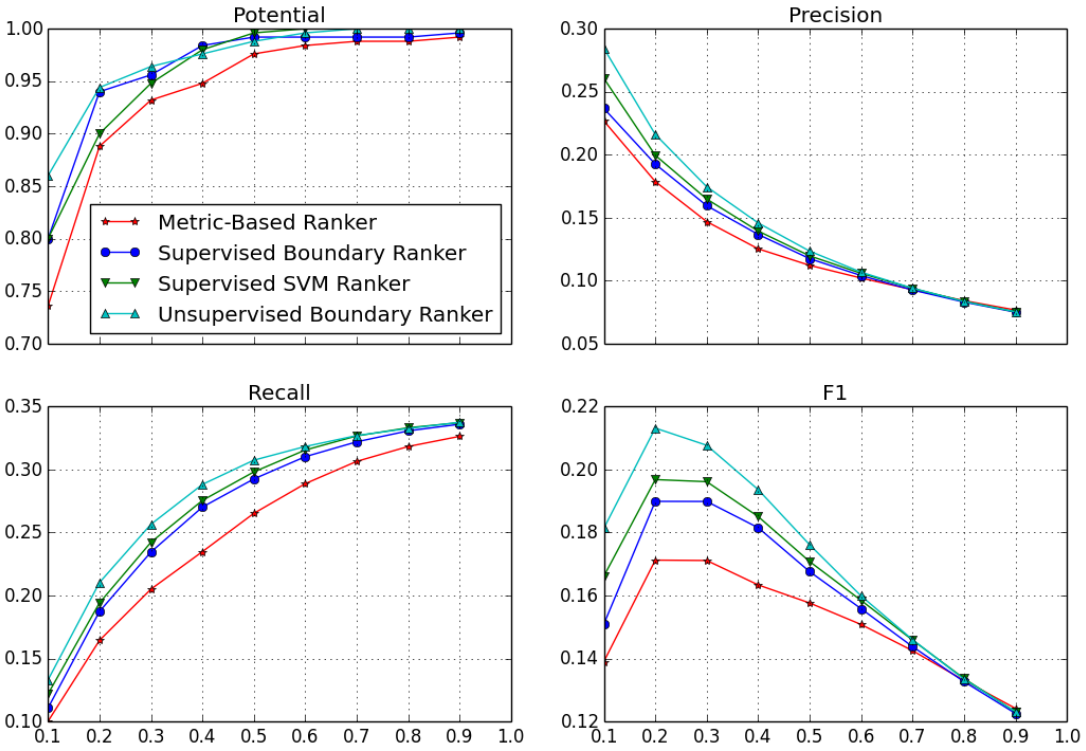

Fig. 7.4 Percentage selection performance results for all generators combined

generator produces a much smaller proportion of spurious candidates than all generators combined. We believe that, because of this difference, the Devlin generator pairs better with a ranking selector that discards only a small proportion of the worst candidates, while the combination of all generators benefits from a selector that discards a much larger proportion of them.

In order to test this hypothesis, we show the performance graphs that result from selecting a fixed integer number of candidate substitutions, rather than a percentage of them. We select from 1 to 15 candidates for each instance in the test set, in intervals of one.

The graphs depicted in Figures 7.5 through 7.8 are in accordance with our hypothesis, and confirm that the ideal number of selected candidates is not identical for all SG approaches, suggesting that the number or percentage of selected candidates should be decided based on the number and type of generator used. While the Devlin, Kauchak and Paetzold generators benefit from a more conservative selector that yield a higher Recall, the combination of all generators requires for a more optimistic selector that yield a higher Precision.

In conclusion, it can be stated that although ranking-based selectors offer more flexibility than disambiguators and classifiers, it is important to investigate the performance and intricacies of the generator being used before determining the number of candidates to select, since the proportion chosen can greatly affect the selector's performance.

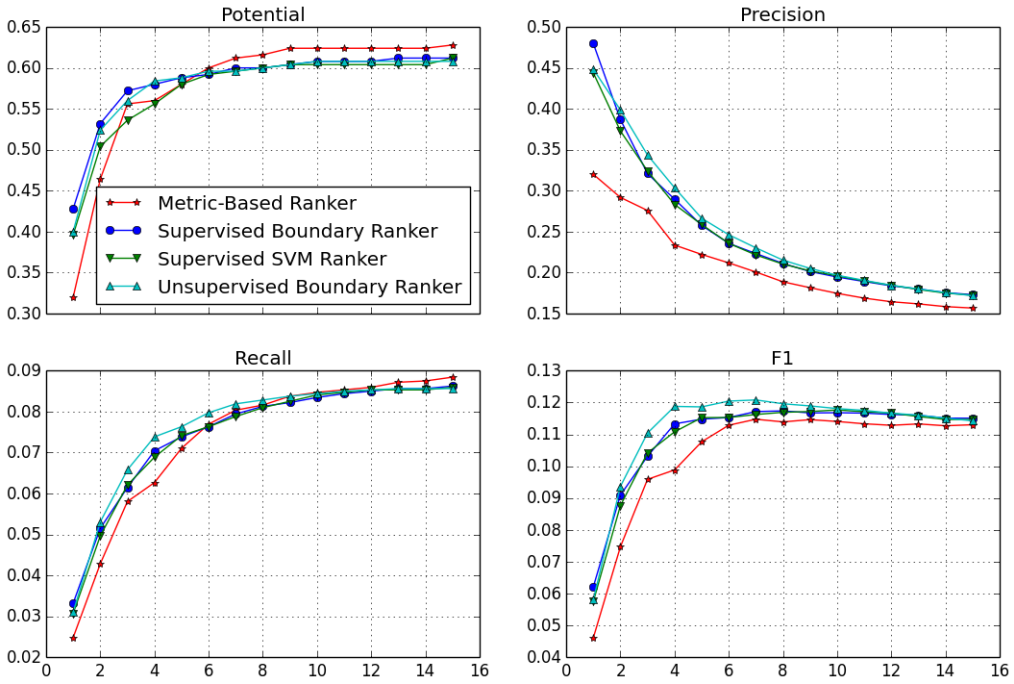

Fig. 7.5 Results for the Devlin generator with respect to fixed numbers of selected candidates

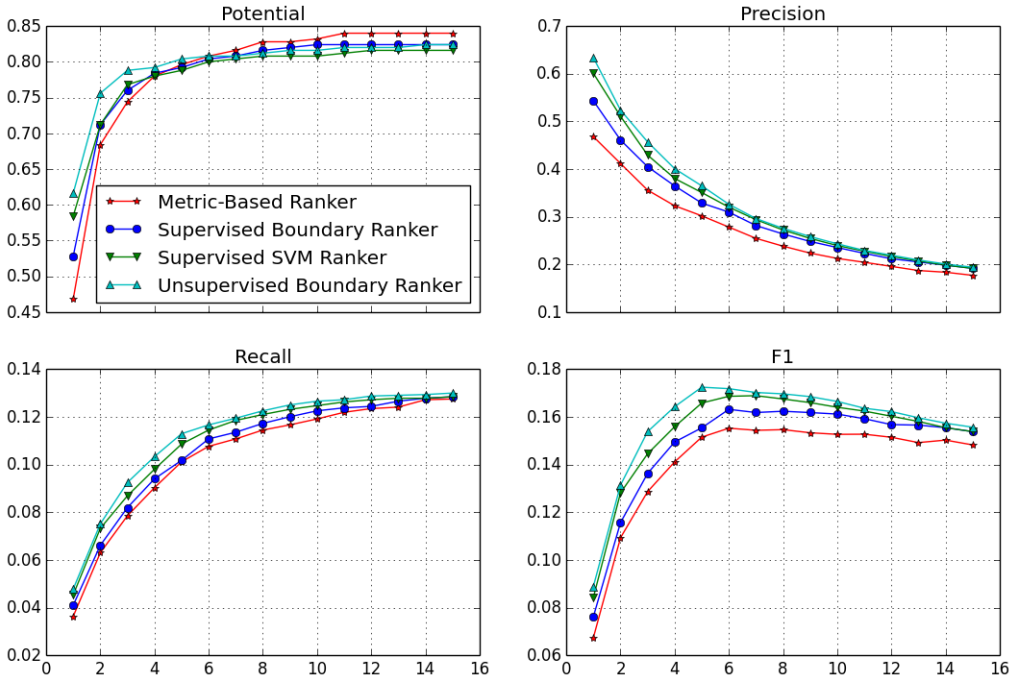

Fig. 7.6 Results for the Kauchak generator with respect to fixed numbers of selected candidates

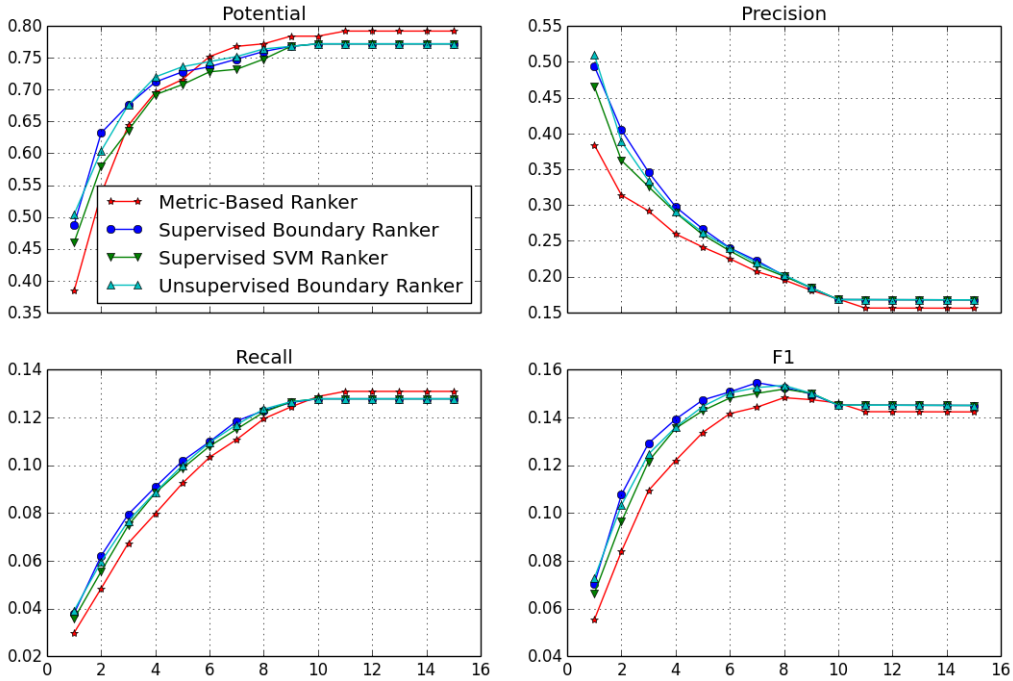

Fig. 7.7 Results for the Paetzold generator with respect to fixed numbers of selected candidates

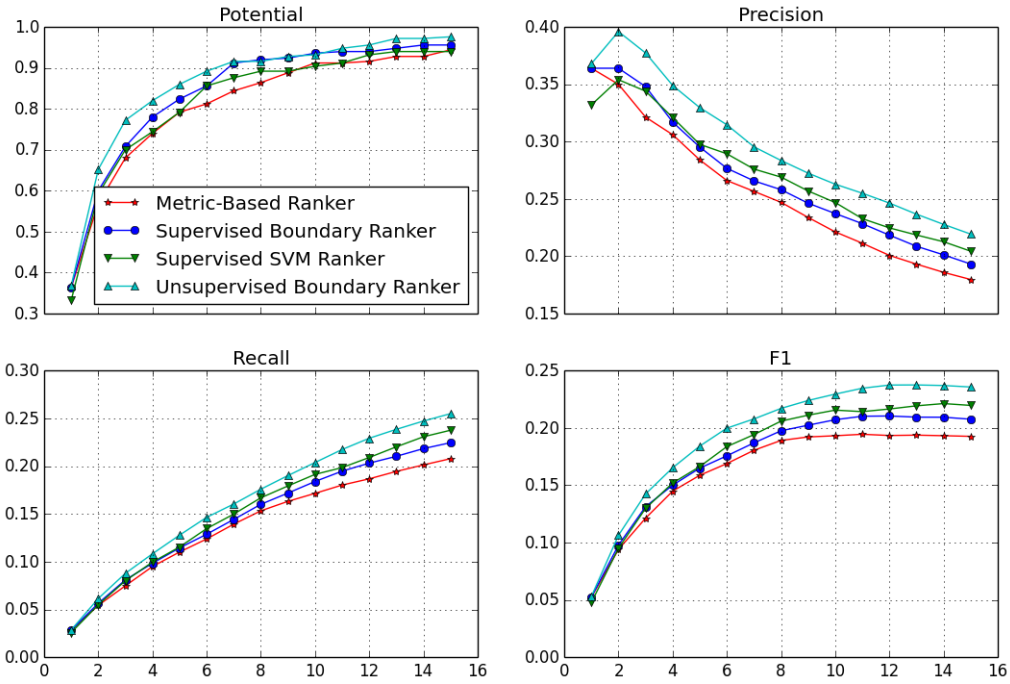

Fig. 7.8 Results for all generators combined with respect to fixed numbers of selected candidates

7.6 Comparing Rankers to Classifiers

Now that we have gathered insight on the relation between the type of generator used and how many candidates should be selected, it is possible to compare the performance of ranking-based selectors to other types of selectors. In this experiment, we compare the performance of the First, Random, Lesk, Path, Biran and Belder selectors, described in Section 7.2.1, and the same four ranking-based selectors from our previous experiment: the Unsupervised Boundary Ranker, Supervised Boundary Ranker, Supervised SVM Ranker and Metric-Based Ranker.

We evaluate the performance of all selectors over the entire Best-First version of LexM-Turk. For the Devlin, Kauchak and Paetzold generators, we select 7 candidate substitutions for each instance in the dataset, and for all generators combined, we select 14. These values were selected based on the best performing selection settings from the graphs in Figures 7.5 through 7.8. The results obtained are illustrated in Tables 7.13 through 7.16.

Selector	Potential	Precision	Recall	F1
First	0.162	0.204	0.017	0.032
Random	0.098	0.141	0.010	0.018
Path	0.140	0.096	0.014	0.024
Lesk	0.140	0.096	0.014	0.024
Biran	0.360	0.127	0.050	0.072
Belder	0.120	**0.447**	0.011	0.022
Metric-Based	0.614	0.197	0.080	0.114
SVM Ranker	0.604	0.229	0.081	0.120
Supervised Boundary	0.610	0.229	0.082	0.120
Unsupervised Boundary	0.610	0.235	0.084	**0.123**
No Selection	**0.616**	0.157	**0.088**	0.113

Table 7.13 Benchmarking SS results for substitutions generated by the Devlin generator

Selector	Potential	Precision	Recall	F1
First	0.076	0.065	0.007	0.012
Random	0.034	0.029	0.003	0.005
Path	0.096	0.076	0.010	0.018
Lesk	0.096	0.076	0.010	0.018
Biran	0.482	0.179	0.075	0.105
Belder	0.228	**0.585**	0.022	0.043
Metric-Based	0.820	0.257	0.114	0.158
SVM Ranker	0.820	0.302	0.123	0.175
Supervised Boundary	0.824	0.298	0.121	0.172
Unsupervised Boundary	0.828	0.307	0.125	**0.177**
No Selection	**0.832**	0.153	**0.134**	0.143

Table 7.14 Benchmarking SS results for substitutions generated by the Kauchak Generator

The scores reveal that the unsupervised boundary ranker is, as suggested in the previous experiments, the current most effective approach for SS. In comparison to not performing selection at all, our approach manages to consistently increase F1 scores while effectively retaining most of the Potential of all generators.

The SS rankers were the only ones able to improve the F1 scores obtained by not performing selection at all. All WSD strategies evaluated, along with any other SS classifiers, have considerably decreased the Potential, Recall and F1 scores of all generators. Although the Belder selector achieved very impressive Precision scores for all instances, this comes at the cost of Potential and Recall.

Selector	Potential	Precision	Recall	F1
First	0.124	0.145	0.012	0.022
Random	0.078	0.094	0.007	0.012
Path	0.224	0.122	0.025	0.041
Lesk	0.224	0.122	0.025	0.041
Biran	0.470	0.187	0.079	0.111
Belder	0.310	**0.335**	0.032	0.058
Metric-Based	0.794	0.220	0.123	0.158
SVM Ranker	0.776	0.296	0.117	0.168
Supervised Boundary	0.788	0.238	**0.132**	**0.170**
Unsupervised Boundary	0.790	0.235	0.131	0.168
No Selection	**0.802**	0.177	0.140	0.156

Table 7.15 Benchmarking SS results for substitutions generated by the Paetzold Generator

Selector	Potential	Precision	Recall	F1
First	0.194	0.052	0.020	0.029
Random	0.116	0.034	0.011	0.017
Path	0.334	0.051	0.043	0.046
Lesk	0.334	0.051	0.043	0.046
Biran	0.602	0.079	0.198	0.113
Belder	0.408	**0.257**	0.046	0.078
Metric-Based	0.934	0.186	0.210	0.197
SVM Ranker	0.952	0.223	0.252	0.237
Supervised Boundary	0.960	0.211	0.239	0.224
Unsupervised Boundary	0.972	0.231	0.261	**0.245**
No Selection	**0.996**	0.071	**0.358**	0.119

Table 7.16 Benchmarking SS results for substitutions generated by all generators combined

7.7 Full Pipeline Evaluation

In our final experiment, we assess the performance of our approach as part of a complete LS system. For that purpose, we combine all pairs of generators and selectors described in Section 7.2.1 with three metric-based Substitution Ranking approaches. The metrics used are:

- **Word Length**: The shorter a word is, the simpler it is.
- **Word Frequency**: The more frequently a word appears in Simple Wikipedia, the simpler it is.
- **Number of Senses**: The more ambiguous a word is, the simpler it is.

To obtain the sense counts we resort to WordNet. The Accuracy scores obtained for all combinations are shown in Tables 7.17 through 7.19.

The SVM ranker did not outperform our unsupervised approach in Substitution Selection alone, but it did perform better in practice. But even though the scores obtained by the SVM ranker are slightly higher than our unsupervised Boundary Ranker overall, F-tests reveal the absence of a statistically significant difference ($p < 0.05$) between them in the settings illustrated in Tables 7.17 and 7.18. In Section 8.5.2 we discuss in more detail the factors we believe to be responsible for the contrast between the results obtained in the Substitution Selection benchmark of Section 7.6 and this full pipeline evaluation.

Nonetheless, all unsupervised and supervised ranking-based selectors have offered consistent improvements over using another selector from literature, or not performing selection at all. It can also be noticed that although supervised Boundary Ranking suffers in performance for some combinations of generators and rankers, our unsupervised Boundary Ranker offers Accuracy scores comparable to that of the SVM ranker in all instances, which is a great achievement, considering that it does not need any manually annotated data to be trained.

7.8 Conclusions

In this Chapter we introduced the concept of ranking-based selectors. Instead of addressing SS as a disambiguation or classification task, in which one must directly decide which candidate substitutions can and cannot replace a given target word, we take it as a ranking problem, in which each and every candidate has a likelihood of being able to correctly replace a target word without compromising its integrity.

	Devlin	Kauchak	Paetzold	All
First	0.112	0.040	0.116	0.048
Random	0.078	0.024	0.068	0.036
Lesk	0.086	0.064	0.146	0.062
Leacock	0.086	0.064	0.120	0.062
Biran	0.124	0.150	0.208	0.082
Brown	0.114	0.216	0.252	0.272
Metric-Based	0.270	0.284	0.348	0.170
Supervised Boundary	0.280	0.280	0.344	0.166
Unsupervised Boundary	0.296	0.328	0.362	0.328
SVM Ranker	0.300	0.336	**0.382**	0.352
No Selection	0.266	0.254	0.324	0.120

Table 7.17 Accuracy scores for candidate substitutions ranked by their length

	Devlin	Kauchak	Paetzold	All
First	0.150	0.076	0.126	0.124
Random	0.092	0.034	0.072	0.074
Lesk	0.110	0.074	0.152	0.118
Leacock	0.110	0.074	0.150	0.118
Biran	0.196	0.228	0.264	0.128
Brown	0.114	0.222	0.270	0.286
Metric-Based	0.332	0.396	0.398	0.220
Supervised Boundary	0.380	0.396	0.430	0.202
Unsupervised Boundary	0.390	0.428	0.436	0.362
SVM Ranker	0.398	**0.442**	0.434	0.372
No Selection	0.248	0.368	0.412	0.114

Table 7.18 Accuracy scores for candidate substitutions ranked by their frequency

	Devlin	Kauchak	Paetzold	All
First	0.132	0.076	0.110	0.124
Random	0.078	0.034	0.072	0.082
Lesk	0.100	0.076	0.146	0.100
Leacock	0.100	0.076	0.122	0.100
Biran	0.152	0.276	0.206	0.138
Brown	0.114	0.214	0.256	0.274
Metric-Based	0.294	0.462	0.322	0.240
Supervised Boundary	0.312	0.480	0.342	0.262
Unsupervised Boundary	0.322	0.480	0.348	0.332
SVM Ranker	0.330	**0.488**	0.382	0.380
No Selection	0.312	0.444	0.328	0.214

Table 7.19 Accuracy scores for candidate substitutions ranked by their number of senses

By combining the Boundary Ranking strategy introduced in Chapter 6 and the Robbins-Sturgeon hypothesis, we introduce a novel, unsupervised ranking approach for Substitution Selection. It learns a model from automatically produced training data by assuming that a target word is irreplaceable, and hence no other candidate substitutions can replace it without compromising the grammaticality and/or changing the meaning of the sentence to some extent.

In our experiments, we discovered that our unsupervised Boundary Ranker can be more effective than various other metric-based and supervised rankers. It can considerably improve the F1 obtained by several generators without compromising their Potential and Recall, despite requiring no manually annotated data for training. We also found that the proportion of candidates selected can greatly influence the performance of an SS ranker, and that it should be determined based on the type of strategy used by the generator. In practice, the ranking-based selectors proved much more effective than any other approaches in literature, and our unsupervised Boundary Ranker obtained performance scores comparable to a much more sophisticated supervised approach.

Given these considerations, we can conclude that ranking-based strategies are a viable approach to Substitution Selection, and our unsupervised Boundary Ranking approach is a flexible, extensible and effective strategy for the task.

Chapter 8

Resources, Tools and Evaluation

Our efforts towards creating more effective LS strategies for non-native English speakers have led to the creation of various useful tools, datasets and corpora for the task. In this Chapter, we describe these resources, as well as a benchmarking and an error analysis experiment where we use them in performance comparisons that outline the main strengths and weaknesses of numerous LS approaches.

More specifically, in what follows, we introduce a structured corpus of subtitles of movies and series for family and children designed to produce frequency counts that better correlate with word familiarity (Section 8.1), a regression bootstrapping algorithm designed to automatically expand a resource with psycholinguistic features of words (Section 8.2), a new framework that provides dozens of LS approaches, including the ones introduced in this thesis (Section 8.3), and a new evaluation dataset for LS that represents the needs of non-native English speakers (Section 8.4). Using our new framework and evaluation dataset, we perform an extensive benchmark that compares the performance of $1,344$ approaches to LS (Section 8.5). We also introduce an automatic error categorisation framework that allows for a more informative analysis of the simplifiers evaluated (Section 8.6).

8.1 SubIMDB: A Corpus of Subtitles

Large corpora of text are certainly one of the most fundamental resources in the field of Computational Linguistics. In Psycholinguistics, it has been long established that word frequencies from corpora play a very important role in cognitive processes. Brysbaert and New (2009a) points out that frequently occurring words are often much more easily perceived, recalled and associated than rare words (Balota and Chumbley, 1984; Rayner and Duffy, 1986). In Text Simplification, researchers have found a strong relationship between frequencies and word simplicity (Devlin and Tait, 1998).

An inherent limitation of work based on word frequency analysis is that the type of resource used as a corpus is often built for a specific communication purpose, such as news (Burgess and Livesay, 1998). This is however not representative of everyday language usage, particularly from a psycholinguistic perspective. The other extreme of the spectrum features resources compiled from user-generated content, such as micro-blogs. However, these resources often suffer from grammar errors and misspellings, excessive use of acronyms and shortenings, partly due to the constrains of the publication means (e.g. limited number of characters) (Pak and Paroubek, 2010).

This is particularly concerning given that previous research has shown that the source from which a corpus was extracted is one of its most important defining traits. Brysbaert and New (2009b), for example, shows that the raw frequencies extracted from a corpus of movie subtitles capture word familiarity more effectively than the ones extracted from other larger corpora. Their corpus corpus was also evaluated on the English Lexical Simplification task of SemEval 2012 by Shardlow (2014b). It helped Shardlow (2014b) achieve scores comparable to those obtained by Google 1T, although it is more than four orders of magnitude smaller. These results are very encouraging for our purposes, since the gold-standard rankings used in their experiments were also produced by non-native English speakers.

Although these findings greatly highlight the potential of spoken language text in LS for non-native English speakers, there are very few examples of resources of this kind available for English. SUBTLEX (Brysbaert and New, 2009b) is a notable exception: it contains texts extracted from $8,388$ subtitles of American movies, and is freely available for download. The OpenSubtitles2016 corpus (Lison and Tiedemann, 2016) is another example, featuring sentences extracted from numerous subtitle files aligned at sentence level across 60 languages. However, since the subtitles in these corpora are not restricted with respect to genre or domain, their proficiency in capturing the everyday language that non-native English speakers are familiar with can be limited. Movies and series span from lighthearted productions for toddlers to historic dramas targeting older audiences, with very distinct vocabulary used.

In an effort to address the lack of reliable everyday language corpora for English, we create SubIMDB, the first structured corpus of subtitles in literature. SubIMDB is composed of subtitles of movies and series written for the "average audience", and can be downloaded in multiple useful formats. In the Sections that follow, we describe the resources and procedures used to build SubIMDB (Section 8.1.1), and evaluate its performance in predicting lexical decision times (Section 8.1.2) and many other psycholinguistic features (Section 8.1.3).

8.1.1 Building SubIMDB

Our goal in creating SubIMDB was to compile and provide freely a large, structured corpus of everyday language that better represents the language which non-native English speakers are familiar with. As a data type, we have chosen subtitles of movies and series, since they are available for dozens of languages. Another advantage of using subtitles as opposed to, for example, chat logs or podcast transcripts, is that movies and series are subject to production standards, and hence the subtitles created for them tend to be composed of linguistically correct constructs.

Acquiring Subtitles

To create a reliable corpus of subtitles one must take into account that movies and series can be of many different genres, and may target very distinct audiences. The compilation of SUBTLEX involved the download of 8,388 subtitles of U.S films and series released between 1900-2007, with no restriction with respect to genre. We took a different approach when creating SubIMDB. We use OpenSubtitles[1] as a data source. One can download subtitles from their API by providing with a production's unique IMDb[2] identifier.

As the first step in creating SubIMDB, we query the IMDb platform searching for identifiers of six types of content: family movies, family series, comedy movies, comedy series, movies for children and series for children. We choose these genres because productions of this kind tend to target viewers of either young or all ages, and hence tend to use accessible language. Our hypothesis is that word usage statistics from this type of content correlate better with psycholinguistic properties of words, such as lexical decision times and age of acquisition.

To obtain the identifiers, we use the IMDb engine[3] to search for and parse all pages under the family and comedy feature film pages, as well as the ones under the family and comedy series categories. Since IMDb does not contain a category specific for children movies and series, we resorted to 15 movies and series lists created by IMDb users to obtain them. In total, we obtained the IMDb identifiers of 9,709 family movies, 8,008 family series, 66,411 comedy movies, 24,776 comedy series, 745 children movies and 124 children series.

We then queried the online OpenSubtitles API for each of these 109,773 IMDb identifiers. Surprisingly, we were only able to find subtitles for 12,618 movies and series. On the other hand, since series are comprised of various episodes, we downloaded subtitles for each

[1] http://www.opensubtitles.org
[2] http://www.imdb.com
[3] http://www.imdb.com/search

episode of every season available in OpenSubtitles. A total of 38,274 subtitles were collected in this way.

Processing Subtitles

In order to make their content more easily accessible, we first tokenized all lines in the subtitles and removed any HTML tags. A filtering algorithm was then applied to discard subtitle lines which:

1. **Refer to metadata or timing indicators**: These lines do not contain meaningful information.
2. **Have more than** 80 **characters**: In most cases, lines with close to or more than 80 characters are composed of sequences of random spurious characters.
3. **Contain advertisement**: These lines refer to credits attributed to the creators of the subtitles in question. Some examples of expressions targeted are "synched by" and "opensubtitles.org".

The resulting corpus contains 228,622,257 words in 39,050,417 lines, which is 4.5 times bigger than SUBTLEX.

8.1.2 Predicting Lexical Decision Times

One of the most popular strategies for frequency norm quality assessment is to evaluate how well they predict lexical decision times. A very popular task in the fields of Psychology and Psycholinguistics, lexical decision, also known as lexical reaction time, refers to the process of deciding whether or not a given sequence of characters is a real word of the language in question (Balota et al., 2007). Previous work has measured the time taken by subjects to make such a decision for certain words, then used correlation metrics to assess how well their frequencies can predict them (Balota et al., 2004; Brysbaert and New, 2009a; Van Heuven et al., 2014; Vega et al., 2011).

In this experiment we assess how well frequencies from different sets of SubIMDB subtitles fair against other well-known corpora in how they correlate with lexical decision times. For this experiment, we extracted word frequencies from various SubIMDB subcorpora:

We compare ours to six frequency norms:

- **KF**: Oldest and most widely used frequency norms, calculated over the Brown corpus (Francis and Kucera, 1979; Rudell, 1993).

All SubIMDB (SubIMDB)	Family movies (SubFAM-M)
All movies (SubMOV)	Family series (SubFAM-S)
All series (SubSER)	Comedy movies (SubCOM-M)
All Family content (SubFAM)	Comedy series (SubCOM-S)
All Comedy content (SubCOM)	Children movies (SubCHI-M)
All children content (SubCHI)	Children series (SubCHI-S)

- **HAL**: *Hyperspace Analogue to Language* word frequency norm, calculated over the HAL corpus, which contains over 131 million words from Usenet newsgroups (Burgess and Livesay, 1998).
- **Wiki**: Word frequencies from Wikipedia, with 97 million words (Kauchak, 2013).
- **SimpleWiki**: Word frequencies from Simple Wikipedia, with 9 million words (Kauchak, 2013).
- **SUBTLEX**: Word frequencies from SUBTLEX, with 51 million words (Brysbaert and New, 2009a).
- **Open2016**: Word frequencies from OpenSubtitles2016, with 2 billion words (Lison and Tiedemann, 2016).

We regularise all norms using Equation 8.1, in which f is the frequency norm value of a word w. This transformation has shown to best represent the relationship between word frequencies and lexical decision times (Balota et al., 2004).

$$norm(f(w)) = \log_{10}(f(w)+1) \tag{8.1}$$

As our test set, we use the MRC psycholinguistic Database (Coltheart, 1981), which provides lexical decision times for $40,468$ words. Like in the experiments of Brysbaert and New (2009a), we consider only the subset of $38,130$ lowercase words in order to avoid most abbreviations and proper nouns.

The correlation scores in Table 8.1 reveal that, while SubIMDB in its entirety yields the highest Spearman (ρ) correlation scores, the SubMOV corpus, which contains only subtitles of movies, yields the highest Pearson (r) correlation. F-tests show a statistically significant difference between frequencies from SubIMDB and all other corpora.

Unlike what was reported in Brysbaert and New (2009a), the HAL norm achieved lower correlation scores than the SUBTLEX norm, despite the fact that the HAL corpus is twice as large as SUBTLEX. This contrast highlights the potential of spoken language corpora in lexical decision prediction.

Norm	Size	ρ	r	F-test
KF	1M	−0.517	−0.486	•••
HAL	131M	−0.641	−0.616	•••
Wiki	97M	−0.531	−0.506	•••
SimpleWiki	9M	−0.560	−0.530	•••
SUBTLEX	62M	−0.653	−0.619	•••
Open2016	2B	−0.657	−0.602	•••
SubIMDB	228M	**−0.659**	−0.624	-
SubMOV	125M	−0.657	**−0.626**	•••
SubSER	100M	−0.652	−0.620	•••
SubFAM	34M	−0.649	−0.614	•••
SubCOM	199M	−0.657	−0.624	•••
SubCHI	17M	−0.634	−0.592	•••
SubFAM-M	17M	−0.640	−0.596	•••
SubFAM-S	17M	−0.632	−0.590	•••
SubCOM-M	107M	−0.655	−0.623	•••
SubCOM-S	91M	−0.651	−0.618	•••
SubCHI-M	8M	−0.625	−0.572	•••
SubCHI-S	8M	−0.606	−0.556	•••

Table 8.1 Lexical decision prediction correlation scores. The last column indicates a statistically significant difference with SubIMDB given $p<0.1$ (•), $p<0.01$ (••) or $p<0.001$ (•••) (F-test).

Our results also indicate a poor performance for the Kucera-Francis coefficient. Despite its use in numerous previous contributions (Brysbaert and New, 2009a; Burgess and Livesay, 1998; Zevin and Seidenberg, 2002), more modern resources proved more effective.

8.1.3 Predicting Psycholinguistic Properties

In addition to lexical decision times, other psycholinguistic properties of words have been studied in terms of their correlation with frequency norms (Paetzold and Specia, 2016c). In this experiment, we evaluate how well the norms described in the previous Section correlate with four psycholinguistic properties extracted from the MRC psycholinguistic Database:

- **Familiarity**: Available for 9,392 words – frequency with which a word is seen, heard or used daily.
- **Age of Acquisition**: Available for 3,503 words – age at which a word is learned.
- **Concreteness**: Available for 8,228 words – how "palpable" the object the word refers to is.

- **Imagery**: Available for 9,240 words – intensity with which a word arouses images.

The results in Table 8.2 reveal that SubFAM-M (family movies) performs better than all other norms in predicting Age of Acquisition and Concreteness, although it is 117 times smaller than OpenSubtitles2016. F-tests reveal a statistically significant difference between SubIMDB and all other corpora.

		Age of Acq.		Familiarity		Concreteness		Imagery	
	Size	r	F-test	r	F-test	r	F-test	r	F-test
KF	1M	−0.447	●●●	0.669	●●●	**−0.180**	●●●	−0.045	●●●
HAL	131M	−0.511	●●●	0.732	●●●	−0.064	●●●	0.086	●●●
Wiki	97M	−0.412	●●●	0.676	●●●	−0.043	●●●	0.084	●●●
SimpleWiki	9M	−0.486	●●●	0.667	●●●	0.011	●●●	0.129	●●●
SUBTLEX	62M	−0.676	●●●	0.774	●●●	0.017	●●●	0.190	●●●
Open2016	2B	−0.666	●●●	**0.799**	●●●	−0.003	●●●	0.185	●●●
SubIMDB	228M	−0.698	-	0.781	-	0.037	-	0.213	-
SubMOV	125M	−0.705	●●●	0.777	●●●	0.031	●●●	0.212	●●●
SubSER	100M	−0.687	●●●	0.777	●●●	0.038	●●●	0.207	●●●
SubFAM	34M	−0.723	●●●	0.758	●●●	0.038	●●●	0.217	●●●
SubCOM	199M	−0.696	●●	0.781	●●●	0.037	●●●	0.211	●●●
SubCHI	17M	−0.709	●●●	0.735	●●●	0.028	●●●	0.201	●●●
SubFAM-M	17M	**−0.746**	●●●	0.742	●●●	0.043	●●●	**0.220**	●●●
SubFAM-S	17M	−0.685	●●●	0.743	●●●	0.007	●●●	0.178	●●●
SubCOM-M	107M	−0.698	●●●	0.777	●●●	0.027	●●●	0.207	●●●
SubCOM-S	91M	−0.690	●●●	0.777	●●●	0.042	●●●	0.209	●●●
SubCHI-M	8M	−0.728	●●●	0.723	●●●	0.026	●●●	0.191	●●●
SubCHI-S	8M	−0.670	●●●	0.704	●●●	−0.006	●●●	0.158	●●●

Table 8.2 Pearson correlation of norms with respect to psycholinguistic properties. Columns following correlation scores indicate a statistically significant difference with SubIMDB given $p<0.1$ (●), $p<0.01$ (●●) or $p<0.001$ (●●●) (F-test).

Perhaps most surprising is the performance of the SubIMDB subset of children movies (SubCHI-M) in predicting Age of Acquisition. Despite its small size, its performance is still much superior than almost all corpora, including OpenSubtitles2016, which is over 250 times larger. Comparing word frequencies from SubCHI-M with the ones in OpenSubtitles2016, we found interesting differences. Table 8.3 shows the most over and underrepresented words in SubCHI-M based on percentages of variance with respect to OpenSubtitles2016.

It can be noticed that while overrepresented words ("*turtles*", "*hedgehog*", etc.) are mostly innocent in nature, underrepresented words describe mostly sexual and/or thought-provoking concepts ("*vagina*", "*abortion*", etc.). These differences reveal that, although subtitle corpora may share traits in general, the domain from which the subtitles are extracted plays an important role. This highlights the often disregarded advantages of a structured, raw

text subtitle corpora like the one we collected here. By making subtitles available in their raw form along with metadata about their source of origin, future research can explore different ways of building the ideal corpus for a given task, e.g. by employing clever subtitle selection and filtering techniques.

	1	2	3	4	5	6	7
Over	hoagy	flintstone	turtles	potter	fantasia	hedgehog	hiccup
Under	vagina	abortion	cartel	intercourse	rapist	overdose	porn

Table 8.3 Representation contrast between the SubCHI-M and OpenSubtitles2016 corpora

In what follows, we describe an approach for the automatic inference of the psycholinguistic features studied in this Section.

8.2 Inferring Psycholinguistic Properties of Words

In the last three decades, much has been found on how the psycholinguistic properties of words influence cognitive processes in the human brain when a subject is presented with either written or spoken forms. A word's Age of Acquisition is an example. The findings in (Carroll and White, 1973) reveal that objects whose names are learned earlier in life can be named faster in later stages of life. Zevin and Seidenberg (2002) show that words learned in early ages are orthographically or phonologically very distinct from those learned in adult life.

Other examples of psycholinguistic properties, such as Familiarity and Concreteness, influence one's proficiency in word recognition and text comprehension. The experiments in (Connine et al., 1990; Morrel-Samuels and Krauss, 1992) show that words with high Familiarity yield lower reaction times in both visual and auditory lexical decision, and require less hand gesticulation in order to be described. Begg and Paivio (1969) found that humans are less sensitive to changes in wording made to sentences with high Concreteness words.

When quantified, these aspects can be used as features for various NLP tasks. Semantic Classification tasks have benefited from the use of such features: by combining Concreteness with other features, Hill and Korhonen (2014) reached the state-of-the-art performance in Semantic Composition (*denotative*/*connotative*) and Semantic Modification (*intersective*/*subsective*) prediction. In LS, the approach of (Jauhar and Specia, 2012) is the only example of approach that exploits such features that we know of. By combining various collocational and psycholinguistic features extracted from the MRC Psycholinguistic Database (Coltheart, 1981), they trained a ranker (Joachims, 2002) that reached first place in the English Lexical Simplification task at SemEval 2012. Nonetheless, their approach could

only explore these psycholinguistic features for a small set of words for which they were available in the MRC database, which would be a problem in a realistic scenario where an LS system is used to simplify thousands of different words from various sources.

Except for the two aforementioned contributions, we were not able to find any other examples of NLP systems that exploit this type of feature. The lack of availability of such resources and the cost inherent to manually estimating psycholinguistic properties are two likely causes for this.

In order to address this problem, we propose a method to automatically infer them. Following the approach of Yarowsky (1995), we train regressors by performing bootstrapping over the MRC Psycholinguistic Database, which provides access to various psycholinguistic properties. To do so, we exploit word embedding models and several other linguistic resources. We compare our approach to various baselines (Section 8.2.3), and incorporate the properties produced in the training of LS systems (Section 8.2.4).

8.2.1 The MRC Psycholinguistic Database

Introduced by Coltheart (1981), the MRC (Machine Readable Dictionary) Psycholinguistic Database is a digital compilation of lexical, morphological and psycholinguistic properties for 150,837 words. The 27 psycholinguistic properties in the resource range from simple frequency measures (Rudell, 1993) to elaborate measures estimated by humans, such as Age of Acquisition and Imagery (Gilhooly and Logie, 1980). However, despite various efforts to populate the MRC Database, these properties are only available for small subsets of the 150,837 words.

We focus on the four manually estimated psycholinguistic properties in the MRC Database described in Section 8.1.3, which are Familiarity, Age of Acquisition, Concreteness and Imagery. We focus on these properties since they have been proven useful and are some of the most scarce in the MRC Database. As we previously discussed, these properties have been successfully used in various approaches for Lexical Simplification and Semantic Classification, and yet are available for no more than 6% of the words in the MRC Database.

8.2.2 Regression Bootstrapping with Word Embeddings

In order to automatically estimate missing psycholinguistic properties in the MRC Database, we resort to bootstrapping. We base our approach on that by Yarowsky (1995), a bootstrapping algorithm which aims to learn a classifier over a reduced set of annotated training instances (or "seeds"). It does so by performing the following five steps:

1. Initialise training set S with the seeds available.

2. Train a classifier over S.
3. Predict values for a set of unlabelled instances U.
4. Add to S all instances from U for which the prediction confidence c is equal or greater than ζ.
5. If at least one instance was added to S, go to step 2, otherwise, return the resulting classifier.

One critical difference between this approach and ours is that our task requires regression algorithms instead of classifiers. In classification, the prediction confidence c is often calculated as the maximum signed distance between an instance and the estimated hyperplanes. There is, however, no analogous confidence estimation technique for regression problems. We address this problem by using word embedding models.

Embedding models have been proved effective in capturing linguistic regularities of words (Mikolov et al., 2013c). In order to exploit these regularities, we assume that the quality of a regressor's prediction on an instance is directly proportional to how similar the instance is to the ones in the labelled set. Since the input for the regressors are words, we compute the similarity between a test word and the words in the labelled dataset as the maximum cosine similarity between the test word's vector and the vectors in the labelled set.

Let M be an embeddings model trained over vocabulary V, S a set of training seeds, ζ a minimum confidence threshold, $sim(w,S,M)$ the maximum cosine similarity between word w and S with respect to model M, R a regression model, and $R(w)$ its prediction for word w. Our bootstrapping algorithm is depicted in Algorithm 2.

Algorithm 2 Regression Bootstrapping

```
1: procedure BOOTSTRAPPING(M, V, S, ζ)
2:     while ||S|| has not converged do
3:         Train R over S
4:         for w ∈ V − S do
5:             if sim(w, S, M) ≥ ζ then Add ⟨w, R(w)⟩ to S
6:             end if
7:         end for
8:     end while
9: end procedure
```

We found that 64,895 out of the 150,837 words in the MRC database were not present in either WordNet or our word embedding models. Since our bootstrappers use features extracted from both these resources, we were only able to predict the Familiarity, Age of Acquisition, Concreteness and Imagery values of the remaining 85,942 words in MRC.

8.2.3 Performance Comparison

In this Section, we assess the performance of our bootstrapping approach in various settings. Since we were not able to find previous work for this task, in these experiments, we compare the performance of our bootstrapping strategy to various baselines. For training, we use the Ridge regression algorithm (Tikhonov, 1963). As features, our regressor uses the word's raw embedding values, along with the following 15 lexical features:

- Word's length and number of syllables, as determined by the Morph Adorner module of LEXenstein (Paetzold and Specia, 2015).
- Word's frequency in the Brown (Francis and Kucera, 1979), SUBTLEX, SubIMDB, Wikipedia and Simple Wikipedia corpora.
- Number of senses, synonyms, hypernyms and hyponyms for word in WordNet.
- Minimum, maximum and average distance between the word's senses in WordNet and the thesaurus' root sense.
- Number of images found for word in the Getty Images database[4].

We train our embedding models using word2vec over a corpus of 7 billion words composed by the SubIMDB corpus, UMBC webbase, News Crawl, SUBTLEX, Wikipedia and Simple Wikipedia. We use 5-fold cross-validation to optimise parameters: ζ, embeddings model architecture (CBOW or Skip-Gram), and word vector size (from 300 to 2,500 in intervals of 200). We include four strong baseline systems in the comparison:

- **Max. Similarity**: Test word is assigned the property value of the closest word in the training set, i.e. the word with the highest cosine similarity according to the word embeddings model.
- **Avg. Similarity**: Test word is assigned the average property value of the n closest words in the training set, i.e. the words with the highest cosine similarity according to the word embeddings model. The value of n is decided through 5-fold cross validation.
- **Simple SVM**: Test word is assigned the property value as predicted by an SVM regressor (Smola and Vapnik, 1997) with a polynomial kernel trained with the 15 aforementioned lexical features.

[4]http://developers.gettyimages.com

- **Simple Ridge**: Test word is assigned the property value as predicted by a Ridge regressor trained with the 15 aforementioned lexical features.

- **Super Ridge**: Identical to Simple Ridge, the only difference being that it also includes the words embeddings in the feature set. We note that this baseline uses the exact same features and regression algorithm as our bootstrapped regressors.

The parameters of all baseline systems are optimised following the same method as with our approach. We also measure the correlation between each of the aforementioned lexical features and the psycholinguistic properties. For each psycholinguistic property, we create a training and a test set by splitting the labelled instances available in the MRC Database in two equally sized portions. All training instances are used as seeds in our approach. As evaluation metrics, we use Spearman's (ρ) and Pearson's (r) correlation. Pearson's correlation is the most important indicator of performance: an effective regressor would predict values that change linearly with a given psycholinguistic property.

The results are illustrated in Table 8.4. While the similarity-based approaches tend to perform well for Concreteness and Imagery, typical regressors capture Familiarity and Age of Acquisition more effectively. Our approach, on the other hand, is consistently superior for all psycholinguistic properties, with both Spearman's and Pearson's correlation scores varying between 0.82 and 0.88. The difference in performance between the Super Ridge baseline and our approach confirm that our bootstrapping algorithm can in fact improve on the performance of a regressor.

The parameters used by our bootstrappers, which are reported below, highlight the importance of parameter optimization in our bootstrapping strategy: its performance peaked with very different configurations for most psycholinguistic properties:

- **Familiarity**: 300 word vector dimensions with a Skip-Gram model, and $\zeta = 0.9$.

- **Age of Acquisition**: 700 word vector dimensions with a CBOW model, and $\zeta = 0.7$.

- **Concreteness**: $1,100$ word vector dimensions with a Skip-Gram model, and $\zeta = 0.7$.

- **Imagery**: $1,100$ word vector dimensions with a Skip-Gram model, and $\zeta = 0.7$.

Interestingly, frequency in the SubIMDB corpus has good linear correlation with Familiarity and Age of Acquisition, much higher than any other feature. For Concreteness and Imagery, on the other hand, the results suggest something different: the further a word is from the root of a thesaurus, the most likely it is to refer to a physical object or entity.

	Familiarity		Age of Acquisition		Concreteness		Imagery	
System	ρ	r	ρ	r	ρ	r	ρ	r
Word Length	-0.238	-0.171	0.501	0.497	-0.170	-0.195	-0.190	-0.193
Syllables	-0.168	-0.114	0.464	0.458	-0.207	-0.238	-0.218	-0.224
Freq: SubIMDB	0.798	**0.725**	**-0.679**	**-0.699**	0.048	0.003	0.208	0.170
Freq: SUBTLEX	**0.827**	0.462	-0.646	-0.251	0.028	0.137	0.187	0.265
Freq: SimpleWiki	0.725	0.488	-0.453	-0.306	0.015	0.145	0.119	0.247
Freq: Wikipedia	0.694	0.283	-0.349	-0.112	-0.076	0.081	0.027	0.134
Freq: Brown	0.706	0.608	-0.380	-0.395	-0.155	-0.214	-0.054	-0.107
Sense Count	0.471	0.363	-0.429	-0.391	0.020	-0.017	0.119	0.059
Synonym Count	0.411	0.336	-0.381	-0.357	-0.036	-0.047	0.070	0.035
Hypernym Count	0.307	0.295	-0.411	-0.387	0.167	0.088	0.268	0.160
Hyponym Count	0.379	0.245	-0.324	-0.196	0.120	0.002	0.196	0.023
Min. Sense Depth	-0.347	-0.072	0.366	0.055	0.151	-0.185	0.127	-0.224
Max. Sense Depth	-0.021	-0.008	-0.197	-0.196	**0.447**	**0.455**	**0.415**	**0.414**
Avg. Sense Depth	-0.295	-0.256	0.215	0.183	0.400	0.428	0.345	0.347
Img. Search Count	0.544	0.145	-0.325	-0.033	-0.037	-0.073	0.117	-0.059
Max. Similarity	0.406	0.402	0.445	0.443	0.742	0.743	0.618	0.605
Avg. Similarity	0.528	0.527	0.536	0.535	0.826	0.819	0.733	0.707
Simple SVM	0.835	0.815	0.778	0.770	0.548	0.477	0.555	0.528
Simple Ridge	0.832	0.815	0.785	0.778	0.603	0.591	0.620	0.613
Super Ridge	**0.847**	**0.833**	**0.827**	**0.820**	**0.859**	**0.852**	**0.813**	**0.800**
Bootstrapping	**0.863**	**0.846**	**0.871**	**0.862**	**0.876**	**0.869**	**0.835**	**0.823**

Table 8.4 Regression correlation scores. In bold are the highest scores within a group (features, baselines, proposed approach), and underlined the highest scores overall.

8.2.4 Extrinsic Evaluation

Here we assess the effectiveness of our bootstrappers in LS. As shown by Jauhar and Specia (2012), psycholinguistic features can help supervised ranking algorithms capture word simplicity. Using the parameters described in Section 8.2.3, we train bootstrappers for these two properties using all instances in the MRC Database as seeds. We then train three rankers with (W) and without (W/O) psycholinguistic features:

- **Horn** (Horn et al., 2014): Uses an SVM ranker trained on various n-gram probability features, as described in Section 5.4.
- **Glavas** (Glavaš and Štajner, 2015): Ranks candidates using various collocational and semantic metrics, as described in Section 5.4.
- **Paetzold**: Ranks words according to a supervised Boundary Ranking approach, with the same best performing configurations used in the experiment of Section 6.6.

We use data from the English Lexical Simplification task of SemEval 2012 to assess systems' performance. The goal of the task is to rank words in different contexts according to their simplicity. The training and test sets contain 300 and 1,710 instances, respectively. We use TRank-at-1, 2 and 3 as our metrics.

The results in Table 8.5 show that the addition of our features lead to performance increases with all rankers. Performing F-tests over the rankings estimated for the simplest candidate in each instance, we have found these differences to be statistically significant ($p < 0.05$). Using our features, the Paetzold Boundary Ranker reaches the best published results for the dataset.

	n=1		*n*=2		*n*=3	
Ranker	W/O	W	W/O	W	W/O	W
Best SemEval	-	0.602	-	-	-	-
Glavas	0.623	**0.636**	0.837	**0.849**	0.929	**0.935**
Horn	0.625	**0.635**	0.843	**0.853**	0.933	**0.942**
Paetzold	0.654	**0.657**	0.858	**0.859**	0.936	**0.939**

Table 8.5 Results on SemEval 2012 LS task dataset

In the Section that follows, we introduce a framework for LS that includes all of the features and approaches described in this experiment, as well as many others.

8.3 LEXenstein: a Framework for Lexical Simplification

Despite the growth in popularity of LS over the years, the absence of tools to support the process of conceiving, creating and evaluating LS systems has been hampering progress in the area. There are very few examples of repositories containing functional implementations of LS systems found in literature. We were only able to find one tool for LS: a set of scripts designed for the training and testing of ranking models provided by Jauhar and Specia (2012)[5]. However, they cover only one step of the process. Because of the scarcity of practical implementations made available, modern contributions that present entire LS approaches, such as the ones of Biran et al. (2011), Bott et al. (2012) and Horn et al. (2014), are often forced to compare their approaches with only simple baselines or at most one other approach in literature.

In an effort to provide a solution for this problem, we developed a new resource for LS. In this Chapter, we introduce LEXenstein: a framework for Lexical Simplification that provides a wide array of resources and utilities for all steps of the LS pipeline. It includes all CWI,

[5]https://github.com/sjauhar/simplex

SG, SS and SR approaches described in this thesis. LEXenstein provides a way for those interested to more easily conceive and evaluate LS systems. We describe LEXenstein in the Sections that follow.

8.3.1 Framework Architecture

LEXenstein is an open-source Python library that provides several approaches for all steps in the Lexical Simplification pipeline. It is distributed under a permissive BSD license and is freely available for download at http://ghpaetzold.github.io/LEXenstein. To increase its flexibility and allow for new content to be more easily incorporated, the library is structured in nine modules:

- **Complex Word Identification**: Contains methods for the identification of complex words in sentences.
- **Substitution Generation**: Contains methods for the generation of substitution candidates for complex words.
- **Substitution Selection**: Contains methods for the selection of which substitutions available can replace an ambiguous complex word in a given sentence.
- **Substitution Ranking**: Contains methods for the ranking of selected candidate substitutions according to their simplicity.
- **Feature Estimation**: Provides a wide range of functions to allow for the user to easily estimate many simplification features for all modules above.
- **Evaluation**: Contains evaluation methods for all steps of the LS pipeline, both jointly and separately.
- **Text Adorning**: Provides an interface for the simplified access of Morph Adorner (Burns, 2013), which can be used for many tasks related to LS.
- **Spelling Correction**: Includes resources for the training of spelling correction models.
- **Utilities**: Offers a series of routines to help the user produce the resources required by other modules.

In order for the input and output produced by the many classes included in LEXenstein to be standardised, we conceived the VICTOR and CWICTOR file formats, which are an elegant way of representing data for both the training and testing of Lexical Simplification

models and systems. They are a reference to Victor Frankenstein, the main character in the Frankenstein novel (Shelley, 2007).

The VICTOR format was conceived to represent datasets for the tasks of Substitution Generation, Selection and Ranking. Each line of a file in VICTOR format is structured as illustrated in Example 8.2, where S_i is the ith sentence in the dataset, w_i a target complex word in the h_ith position of S_i, c_i^j a substitution candidate and r_i^j its simplicity ranking.

$$\langle \mathrm{S}_i \rangle \ \langle \mathrm{w}_i \rangle \ \langle \mathrm{h}_i \rangle \ \langle \mathrm{r}_i^1 : \mathrm{c}_i^1 \rangle \ \langle \mathrm{r}_i^2 : \mathrm{c}_i^2 \rangle \cdots \langle \mathrm{r}_i^{n-1} : \mathrm{c}_i^{n-1} \rangle \ \langle \mathrm{r}_i^n : \mathrm{c}_i^n \rangle \tag{8.2}$$

Each bracketed component in the example above is separated by a tabulation mark.

The CWICTOR format was conceived to represent datasets for the task of Complex Word Identification. Each line of a file in CWICTOR format is structured as illustrated in Example 8.3, where S_i is the ith sentence in the dataset, w_i a the word in the h_ith position of S_i, and l_i a binary label, which must have value 1 if w_i is complex, and value 0 otherwise.

$$\langle \mathrm{S}_i \rangle \ \langle \mathrm{w}_i \rangle \ \langle \mathrm{h}_i \rangle \ \langle \mathrm{l}_i \rangle \tag{8.3}$$

Each bracketed component in the example above is separated by a tabulation mark.

8.3.2 Complex Word Identification

LEXenstein's Complex Word Identification module supports five distinct approaches. All of them require as input a dataset in CWICTOR format, and produce as output one label for each instance in it. Each approach is represented by one of the following Python classes:

- **SimplifyNoneIdentifier**: Assumes that no words should be simplified, and consequently predicts that none of the words are complex.
- **SimplifyAllIdentifier**: Assumes that all words should be simplified, and consequently predicts that all words are complex.
- **LexiconIdentifier**: Employs a lexicon-based approach for CWI, as described in Section 2.2.3. It assumes that if a given word w is present in lexicon V, then it is complex/simple. This class takes as input a lexicon file along with a label, which must have value "complex" if the lexicon refers to a list of complex words, or "simple" if it refers to a lexicon of simple words.
- **ThresholdIdentifier**: Employs a threshold-based approach for CWI, as described in Section 2.2.2. It assumes that, given a word w and a metric $M(w)$, there exists an

optimal threshold t which best separates complex and simple words. This class requires for a set of training instances over which threshold t is to be estimated. By using a brute force search algorithm, it estimates threshold t over the training set, and uses it it to judge whether or not a given word is complex.

- **MachineLearningIdentifier**: Employs Machine Learning techniques in the learning of CWI models, as described in Section 2.2.5. Given a set of training instances and a specified learning technique, the class fits a model over the data, and then uses it to classify new words into complex or simple. It supports seven Machine Learning techniques:

  1. Support Vector Machines
  2. Passive Aggressive Learning
  3. Perceptron
  4. Decision Trees
  5. Adaptive Boosting
  6. Gradient Boosting
  7. Random Forests

  It allows also for one to combine multiple identifiers using Hard Voting, Soft Voting and Performance-Oriented Soft Voting, as described in Section 3.1.5. This class resorts to the scikit-learn library Pedregosa et al. (2011), which allows for one to easily add a new Machine Learning technique to the LEXenstein framework if needed.

8.3.3 Substitution Generation

LEXenstein's Substitution Generation module offers support for seven approaches. They require as input a dataset in VICTOR format, and produce candidate substitutions for each target word in the dataset. Each SG approach is represented by one of the following Python classes:

- **DevlinGenerator**: Employs the strategy of Devlin and Tait (1998), that extracts synonyms from WordNet as described in Section 5.4.

- **KauchakGenerator**: Employs the strategy of Horn et al. (2014), that uses complex-to-simple parallel corpora as described in Section 5.4.

- **BiranGenerator**: Employs the strategy of Biran et al. (2011), that uses cartesian products between complex and simple data as described in Section 5.4.

- **YamamotoGenerator**: Employs the strategy of Kajiwara et al. (2013), that uses dictionary definitions as described in Section 5.4.

- **GlavasGenerator**: Employs the strategy of Glavaš and Štajner (2015), that uses traditional word embedding models as described in Section 5.4.

- **MerriamGenerator**: Extracts a dictionary linking words to their synonyms, as listed in the Merriam Thesaurus.

- **PaetzoldGenerator**: Employs the strategy described in Chapter 5, which extracts substitutions from a context-aware word embeddings model. It requires a context-aware word embedding model in binary format, trained with word2vec over an annotated corpus. It supports multiple POS tag annotation formats.

8.3.4 Substitution Selection

LEXenstein's Substitution Selection module provides eight approaches. All of them require as input a dataset in the VICTOR format, and either a dictionary of substitutions produced by a class from the Substitution Generation module, or a list of selected substitutions produced by another selector. This allows the user to combine multiple selectors in the pipeline. As output, they produce a set of selected substitutions for each entry in the VICTOR dataset.

Each approach in the SS module is represented by one of the following Python classes:

- **VoidSelector**: Performs no filtering, selecting all generated substitutions for each target word.

- **BelderSelector**: Employs the word clustering strategy of De Belder and Moens (2010), as described in Section 5.4.

- **BiranSelector**: Employs the co-occurrence model filtering strategy of Biran et al. (2011), as described in Section 5.4.

- **WSDSelector**: Allows for the user to use classic WSD approaches such as the Lesk (Lesk, 1986) and Path Similarity Wu and Palmer (1994) algorithms, as described in Section 5.4.

- **WordVectorSelector**: Employs the strategy of Paetzold and Specia (2015), in which a word vector model is used to determine which substitutions have the closest meaning to that of the sentence being simplified. It requires a binary word vector model produced by word2vec, and can be configured in many distinct ways. It retrieves a user-defined percentage of the substitutions, which are ranked with respect to the cosine distance between its word vector and the sum of some or all of the sentences' words, depending on the settings defined by the user.

- **BoundarySelector**: Employs the Boundary Ranking selection strategy described in Chapter 7. It supports various linear models. This approach can be trained in both supervised and unsupervised fashion.

- **SVMBoundarySelector**: Employs an adapted version of the Boundary Ranking selection strategy described in Chapter 7 which allows for non-linear ranking models to be used. This approach can be trained in both supervised and unsupervised fashion.

- **SVMRankSelector**: Employs the supervised SVM ranking selection strategy described in Section 7.3.

8.3.5 Substitution Ranking

LEXenstein's Substitution Ranking module provides eight approaches. All of them receive as input datasets in the VICTOR format, which can be either training/testing datasets already containing only valid substitutions in context, or datasets generated with (potentially noisy) substitutions by a given SS approach. Each approach in the SR module is represented by one of the following Python classes:

- **MetricRanker**: Employs a simple strategy based on the values of a single feature provided by the user, extracted as described in Section 8.3.6.

- **BiranRanker**: Employs the metric-based strategy introduced by Biran et al. (2011), described in the Section 2.5.2. It ranks candidates according to the metric illustrated in Equation 8.4, in which $F(c,C)$ is the frequency of candidate c in corpus C, and $\|c\|$ the length of candidate c.

$$M(c) = \frac{F(c,\text{Complex})}{F(c,\text{Simple})} \times \|c\| \tag{8.4}$$

- **YamamotoRanker**: Employs the elaborate metric-based strategy introduced by Kajiwara et al. (2013), described in Section 2.5.2. Their metric is described in Equation 8.5,

and models simplicity as the combination of features such as n-gram frequencies, word co-occurrence model similarity measures and semantic distance.

$$\begin{aligned} M(S,w_t,w_c) = & \alpha_1 F_{corpus}(w_c) + \\ & \alpha_2 Sense(w_c,w_t) + \\ & \alpha_3 Cooc(w_c,S) + \\ & \alpha_4 Log(w_c,S) + \\ & \alpha_5 Trigram(w_c,S) + \\ & \alpha_6 Sim(w_c,w_t) \end{aligned} \tag{8.5}$$

More details about this strategy can be found in (Kajiwara et al., 2013).

- **BottRanker**: Employs the metric-based strategy introduced by Bott et al. (2012), described in the Section 2.5.2. It ranks candidates according to their length and frequency in a corpus. Their metric is described in Equations 8.7 and 8.6, where w_c is a substitution candidate for a target complex word, and $F_{simple}(w_c)$ its frequency in a corpus of simple text.

$$score_{wl}(w_c) = \begin{cases} \sqrt{\|w_c\| - 4} & \textbf{if } \|w_c\| \geq 5 \\ 0 & \textbf{otherwise} \end{cases} \tag{8.6}$$

$$score_{freq}(w_c) = \log\left(F_{simple}(w_c)\right) \tag{8.7}$$

- **SVMRanker**: Use the SVMRank approach of Joachims (2002), described in Section 6.3.4.

- **GlavasRanker**: Employs the rank averaging strategy of Glavaš and Štajner (2015), described in Section 6.3.4.

- **BoundaryRanker**: Employs the supervised Boundary Ranking strategy with linear models presented in Chapter 6.

- **SVMBoundaryRanker**: Employs the same Boundary Ranking approach, but allows for non-linear models with SVMs.

8.3.6 Feature Estimation

LEXenstein's Feature Estimation module allows the calculation of several features for LS-related tasks. Its class FeatureEstimator allows the user to select and configure many features commonly used by the aforementioned strategies for CWI, SG, SS and SR.

The FeatureEstimator object can be used either for the creation of LEXenstein's generators, selectors and rankers, or in stand-alone setups. For the latter, the class provides a function called *calculateFeatures*, which produces a matrix $M \times N$ containing M feature values for each of the N substitution candidates listed in the dataset. Each of the current 43 features supported must be configured individually. They can be grouped in eight categories:

- **Lexicon-oriented**: A binary feature that receives value 1 if a candidate appears in a given vocabulary, and 0 otherwise.
- **Morphological**: Features that exploit morphological characteristics of substitutions. They include:
  1. The length of a candidate.
  2. The number of syllables of a candidate.
- **Collocational**: Comprised of several raw frequency counts and language model probabilities of the form $P\left(S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}\right)$, where c is a candidate substitution in the hth position in sentence S, and S_{h-l}^{h-1} and S_{h+1}^{h+r} are n-grams of size l and r, respectively. The features in this category are:
  1. The language model probability of the entire sentence S with the target word replace by a candidate.
  2. The language model probability of an n-gram.
  3. The raw frequency count of an n-gram in a corpus.
  4. A binary feature that receives value 1 if an n-gram is in a corpus, and 0 otherwise.
- **Pop-Collocational**: Comprised of several raw frequency counts and language model probabilities of the "pop" n-grams introduced by Jauhar and Specia (2012). They differ from typical collocational features in the sense that, instead of retrieving the frequency of the n-gram itself, it retrieves the highest frequency between three variants of an n-gram: its original form, $S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}$, its leftmost "popped" version, $S_{h-l}^{h-2}\, c\, S_{h+1}^{h+r}$, its rightmost "popped" version, $S_{h-l}^{h-1}\, c\, S_{h+2}^{h+r}$, and its double "popped" version, $S_{h-l}^{h-2}\, c\, S_{h+2}^{h+r}$. The features in this category are:

1. The language model probability of a pop n-gram.
2. The raw frequency count of a pop n-gram in a corpus.

- **Tagged-Collocational**: Comprised of several raw frequency counts of "tagged" n-grams. They differ from typical collocational features in the sense that instead of retrieving the frequency of an n-gram in its original form, $S_{h-l}^{h-1}\, c\, S_{h+1}^{h+r}$, it retrieves the frequency of a candidate surrounded by the neighbor words' POS tags, $P_{h-l}^{h-1}\, c\, P_{h+1}^{h+r}$, given that P is the set of POS tags that describe sentence S. The features in this category are:

  1. The raw frequency count of a tagged n-gram in a corpus.
  2. A binary feature that receives value 1 if a tagged n-gram is in a corpus, and 0 otherwise.

- **Sense-oriented**: Comprised of features that describe the semantic information of a candidate substitution. They are:

  1. The number of senses of a candidate.
  2. The number of synonyms of a candidate.
  3. The number of hypernyms of a candidate.
  4. The number of hyponyms of a candidate.
  5. The maximum semantic distance between all of a candidate's senses in a thesaurus.
  6. The minimum semantic distance between all of a candidate's senses in a thesaurus.
  7. A binary feature that receives value 1 if the candidate is a synonym of the target word, and 0 otherwise.
  8. A binary feature that receives value 1 if the candidate is a hypernym of the target word, and 0 otherwise.
  9. A binary feature that receives value 1 if the candidate is a hyponym of the target word, and 0 otherwise.

- **Syntactic**: Comprised of features that measure how likely a candidate substitution is of assuming the syntactic role of the target word. We describe the syntactic role of a target word t in sentence S as its POS tag and its set of subject dependency links, $(t \rightarrow S_i)$, and object dependency links, $(t \leftarrow S_i)$. The features in this category are:

1. The conditional probability of the candidate assuming the POS tag of the target word, as described in Section 7.2.2.
2. The average probability of the candidate assuming the subject dependency links of the target word.
3. The average raw occurrence counts in which the candidate assumes the subject dependency links of the target word.
4. A binary feature that receives value 1 if the candidate assumes all subject dependency links of the target word at least once in a corpus.
5. The average probability of the candidate assuming the object dependency links of the target word.
6. The average raw occurrence counts in which the candidate assumes the object dependency links of the target word.
7. A binary feature that receives value 1 if the candidate assumes all object dependency links of the target word at least once in a corpus.
8. The average probability of the candidate assuming all dependency links of the target word.
9. The average raw occurrence counts in which the candidate assumes all dependency links of the target word.
10. A binary feature that receives value 1 if the candidate assumes all dependency links of the target word at least once in a corpus.

- **Semantic**: Comprised of features that describe not only the semantic content pertaining to a candidate individually, but also its semantic similarity with the target word and the sentence in question. The features in this category are:

  1. The probability of the target word being translated into a given candidate.
  2. The candidate's word embedding values, as determined by a word embeddings model.
  3. The cosine similarity between the candidate's and the target word's embedding vectors.
  4. The average cosine similarity between the candidate's embeddings vector, and those of all content words in the sentence.
  5. The candidate's word embedding values, as determined by a context-aware word embeddings model, as described in Section 5.2.

6. The cosine similarity between the candidate's and the target word's a context-aware embedding vectors.
7. The average cosine similarity between the candidate's a context-aware embeddings vector, and those of all content words in the sentence.

- **Psycholinguistic**: Comprised of the bootstrapped psycholinguistic features described in Section 8.1.3. The features in this category are:

1. Familiarity
2. Age of Acquisition
3. Concreteness
4. Imagery

Notice that each of these features require different resources, such as language models, n-gram count tables, vocabularies, word embedding models, etc. Information such as synonyms and hypernyms are extracted from WordNet.

8.3.7 Evaluation

Since one of the goals of LEXenstein is to facilitate the benchmarking LS approaches, it is crucial that it provides evaluation methods. This module includes methods for the evaluation of all pipeline steps, both individually and in combination. These are the same methods used in the experiments of Chapters 3, 5, 6, and 7. It contains four classes:

- **IdentifierEvaluator**: Provides the evaluation metrics for CWI methods described in Section 3.1.4. It requires a gold-standard in the CWICTOR format and a set of predicted binary labels. It returns the Accuracy, Precision, Recall, F-score and G-score:

  - **Accuracy**: The proportion of correctly predicted labels.
  - **Precision**: The proportion of correctly predicted positive labels.
  - **Recall**: The proportion of gold-standard positive labels that were correctly predicted.
  - **F-score**: The harmonic mean between Precision and Recall.
  - **G-score**: The harmonic mean between Accuracy and Recall, as described in Section 3.1.4.

- **GeneratorEvaluator**: Provides the evaluation metrics for SG methods described in Section 5.4. It requires a gold-standard in the VICTOR format and a set of generated substitutions. It returns the Potential, Precision, Recall and F1:
  - **Potential**: The proportion of instances in which at least one of the substitutions generated is present in the gold-standard.
  - **Precision**: The proportion of generated substitutions that are present in the gold-standard.
  - **Recall**: The proportion of gold-standard substitutions that are included in the generated substitutions.
  - **F1**: The harmonic mean between Precision and Recall.
- **SelectorEvaluator**: Provides the evaluation metrics for SS methods described in Section 5.4. It requires a gold-standard in the VICTOR format and a set of selected substitutions. It returns the Potential, Precision, Recall and F1 of the SS approach, as defined above.
- **RankerEvaluator**: Provides evaluation metrics for SR methods. It requires a gold-standard in the VICTOR format and a set of ranked substitutions. It returns TRank-at-1:3, described in Section 6, and Recall-at-1:3, described in (Specia et al., 2012):
  - **Trank-at-n**: The proportion of instances in which a candidate of gold-rank $r \leq n$ was ranked first.
  - **Recall-at-n**: The proportion of candidates of gold-rank $r \leq n$ that are ranked in positions $p \leq n$.
- **PipelineEvaluator**: Provides the evaluation metrics for the entire LS pipeline described in Section 5.4. It requires as input a gold-standard in the VICTOR format and a set of ranked substitutions which have been generated and selected by a given set of approaches. It returns the approaches' Precision, Accuracy and Change Proportion:
  - **Precision**: The proportion of instances in which the highest ranking substitution is either the target complex word itself or is in the gold-standard.
  - **Accuracy**: The proportion of instances in which the highest ranking substitution is not the target complex word itself and is in the gold-standard.
  - **Changed Proportion**: The proportion of instances in which the highest ranking substitution is not the target complex word itself.

- **PLUMBErr**: Performs an error identification analysis of a simplifier. It uses the strategy described in Chapter 8.6 to identify and categorize the errors made by a simplifier. The error categories identified are, as described in Section 8.6.1:

  - **Type 1**: No error. The system did not commit any mistakes while simplifying the target word.
  - **Type 2A**: The system mistook the target complex word for simple.
  - **Type 2B**: The system mistook the target simple word for complex.
  - **Type 3A**: The system did not produce any candidate substitutions for the target word.
  - **Type 3B**: The system did not produce any simpler candidate substitutions for the target word.
  - **Type 4**: The system replaced the target word with a candidate that compromises the integrity of the sentence.
  - **Type 5**: The system replaced the target word with a candidate that does not simplify the sentence.

8.3.8 Utilities

LEXenstein also provides various modules with useful tools that can be used not only in LS, but also in many other NLP tasks.

The Text Adorning module provides a Python interface to the Morph Adorner Toolkit (Paetzold, 2015a), a set of Java tools that facilitates the access to several of Morph Adorner's functionalities, such as:

- Word lemmatisation
- Word stemming
- Syllable splitting
- Noun inflection
- Verb tensing
- Verb conjugation

It also includes a functionality not supported by Morph Adorner: adjective and adverb inflection. It allows one to inflect an adjective or adverb to its superlative or comparative tense.

The Spelling Correction module includes a class that allows for one to both train and use the spelling correction model proposed by Norvig[6]. The user can train the model over any corpus of text, save it in binary format, and then use it to correct words whenever necessary.

LEXenstein also includes a dedicated Utilities module that offers several functions that help the user produce files and resources required by identifiers, generators, selectors and rankers. Among those functions are:

- Dependency parsing
- Tag generalisation
- Tagged n-grams file creation
- N-grams file filtering
- Vocabulary extraction
- Translation probabilities file creation
- POS tag conditional probability model training

In the following Section, we introduce a new evaluation dataset for LS and showcase the potential of LEXenstein by conducting a benchmarking of LS systems.

8.4 NNSeval: A New Dataset for Lexical Simplification

To facilitate the evaluation of LS systems, the works of De Belder and Moens (2012a) and Horn et al. (2014) introduce two resources for the task: the LSeval and LexMTurk datasets. The instances in both datasets, 930 in total, are composed of a sentence, a target word, and candidate substitutions ranked by simplicity. Using different metrics, one is able to evaluate each step of the LS pipeline over these datasets, both individually and jointly.

There is, however, no way of knowing the profile of the annotators who produced these datasets. In both datasets, the candidate substitutions were suggested and ranked by English speakers from the U.S, who are unlikely to be non-native speakers of English in their majority. This limitation renders these datasets unsuitable for the evaluation of our approach because

[6]http://norvig.com/spell-correct.html

i) the target words used may not be considered complex by non-native speakers of English ii) the candidate substitutions suggested may be deemed complex by non-native speakers of English. In order to reuse these resources and create a more reliable dataset, we have filtered them according to the data acquired in the CWI user study described in Section 3.1.

To create our dataset, we have first pre-processed the instances of both datasets. Inspecting LSeval, we were able to notice that most candidate substitutions for verbs and nouns did not share the same inflection as the target word. Table 8.6 illustrates some examples of such instances. We solve this problem by using the Text Adorning module of LEXenstein to inflect all candidate verbs and nouns in the LSeval dataset to the same morphological stance as the target word. Table 8.7 shows the corrected version of the instances in Table 8.6.

Sentence	Target	Candidates
There are different types of managed care systems.	managed	control, manage, coordinate, administer
The GARCH model is an infinite order arch model with a geometrically declining set of weights.	declining	fall, reduce, decline, decrease
Even galls that formed on their trunks were eaten.	galls	abnormal growth, gall, pustule, grub
But he could feel the whale was sensing him with sound pulses.	pulses	wave, vibration, beat, throb, pulse
That said, the best foundations are focused on accomplishing programmatic missions.	missions	aim, task, project, operation

Table 8.6 Incorrectly inflected candidate substitutions from the LSeval dataset

Sentence	Target	Candidates
There are different types of managed care systems.	managed	controled, managed, coordinated, administered
The GARCH model is an infinite order arch model with a geometrically declining set of weights.	declining	falling, reducing, declining, decreasing
Even galls that formed on their trunks were eaten.	galls	abnormal growth, galls, pustules, grubs
But he could feel the whale was sensing him with sound pulses.	pulses	waves, vibrations, beats, throbs, pulses
That said, the best foundations are focused on accomplishing programmatic missions.	missions	aims, tasks, projects, operations

Table 8.7 Correctly inflected candidate substitutions from the LSeval dataset

We have also noticed that some of the candidates in the LexMTurk dataset had spelling errors. Table 8.8 illustrates some examples. To address this, we used the Spelling Correction module of LEXenstein. The spelling-corrected versions of the instances in Table 8.8 are illustrated in Table 8.9.

Sentence	Target	Candidates
It has an attached tag with a unique "Secret Code" printed on it.	unique	sepcial, seperate
Hurricane-force wind gusts were reported in New England.	hurricane-force	very stong, stong
Simon Phillip Cowell is an English music executive, television producer and entrepreneur.	entrepreneur	buiness, businesman
Techno is a form of electronic dance music that emerged in Detroit.	emerged	came frome, noticable
It is also known as a clothes iron, flat iron, or smoothing iron.	smoothing	levelling, flatning

Table 8.8 Incorrectly spelled candidate substitutions from the LexMTurk dataset

Sentence	Target	Candidates
It has an attached tag with a unique "Secret Code" printed on it.	unique	special, separate
Hurricane-force wind gusts were reported in New England.	hurricane-force	very strong, strong
Simon Phillip Cowell is an English music executive, television producer and entrepreneur.	entrepreneur	business, businessman
Techno is a form of electronic dance music that emerged in Detroit.	emerged	came from, noticeable
It is also known as a clothes iron, flat iron, or smoothing iron.	smoothing	leveling, flattening

Table 8.9 Correctly spelled candidate substitutions from the LexMTurk dataset

In sequence, we removed all candidates which were deemed complex by at least one annotator in our user study. Finally, we have discarded all instances in which the target word was not deemed complex by any of the annotators. As a result, we have compiled a dataset of 239 instances, which we name NNSeval. In what follows, we use NNSeval to benchmark hundreds of LS approaches.

8.5 Benchmarking Lexical Simplification Systems for Non-Native Speakers

Although there are literature surveys on Text Simplification that describe the most effective LS approaches (Shardlow, 2014b; Siddharthan, 2014), they do not provide performance comparisons between them. We believe that the reason for this is the absence of software made available by the creators of LS systems.

The only examples of performance comparisons in Lexical Simplification literature are the ones conducted by Specia et al. (2012), Shardlow (2013a) and (Paetzold and Specia, 2016f). Specia et al. (2012) report the findings of the English Lexical Simplification task of SemEval 2012, one of the most frequently cited contributions to LS, in which several rankers were evaluated over a Substitution Ranking dataset annotated by non-native English speakers. The work of Shardlow (2013a) focuses on the task of Complex Word Identification. They introduce a new automatically generated dataset for the task, and compare a threshold-based approach that separates complex words with respect to their word frequencies, an SVM approach trained with six features related to word complexity, and a "simplify everything" approach. The Complex Word Identification task of SemEval 2016 (Paetzold and Specia, 2016f), described in Section 3.1.5, also focuses on CWI, but includes a much more substantial benchmark of 42 systems submitted by participants along with various baselines.

Performance comparisons allow a better understanding of the current state-of-the-art of a task and the information they provide can influence how researchers approach a given task in order to find more effective approaches for it. In this Section we present the first benchmarking of LS approaches in literature. Using the previously introduced NNSeval dataset as a gold-standard, we conduct performance comparisons for various steps of LS pipeline, both individually, and jointly. The entire benchmark was conducted using the methods and classes provided by the LEXenstein framework.

8.5.1 Performance Comparisons

In this Section, we present the results obtained in our benchmarks. We evaluate all steps of the pipeline except CWI, for which a benchmark has already been provided in Sections 3.1.4 and 3.1.5. CWI is also excluded from our full pipeline evaluation because there are no datasets available that allow us to jointly evaluate all steps of the pipeline. We choose instead to use the usual full pipeline evaluation method employed in the performance comparisons of Chapters 5 and 7, which is a consolidated technique that combines Substitution Generation, Selection and Ranking (Glavaš and Štajner, 2015; Horn et al., 2014). In Chapter 8.6 we

introduce a new evaluation method for LS systems that addresses this limitation and allows for CWI to be jointly benchmarked along with the remaining steps of the pipeline.

Substitution Generation

In this benchmark we evaluate the performance of various Substitution Generation strategies in LEXenstein. We also evaluate the performance of all generators combined, by creating a super set of the output produced by all of them.

Settings We configure the generators of LEXenstein as follows:

- **Devlin**: Uses the same configuration described in Section 5.4.1.
- **Kauchak**: Uses the same configuration described in Section 5.4.1.
- **Biran**: Uses the same configuration described in Section 5.4.1.
- **Yamamoto**: Uses the same configuration described in Section 5.4.1.
- **Merriam**: Uses the Merriam Thesaurus[7] as a source for synonyms.
- **Glavas**: Selects 10 candidates for each target word. We use the word2vec toolkit to train the word embeddings model used. The corpus used contains 7 billion words, and includes the SubIMDB corpus, UMBC webbase, News Crawl, SUBTLEX, Wikipedia and Simple Wikipedia. As training parameters, we use the bag-of-words model (CBOW), and $1,300$ dimensions for the embedding vectors.
- **Paetzold**: Selects 10 candidates for each target word. Uses the same tools, and resources used by the Glavas generator except for the retrofitted context-aware word embeddings model, which is trained over a corpus annotated with universal tags produced by the Stanford Parser, and retrofitted over WordNet 3.0. As training parameters, we use the bag-of-words model (CBOW), and $1,300$ dimensions for the word vectors.

Datasets For testing, we use the NNSeval dataset, described in Section 8.4.

[7] http://www.dictionaryapi.com/products/api-collegiate-thesaurus.htm

Evaluation Metrics For evaluation, we use the metrics described in Section 8.3.7, which are:

- **Potential**: The proportion of instances in which at least one of the substitutions generated is present in the gold-standard.
- **Precision**: The proportion of generated substitutions that are present in the gold-standard.
- **Recall**: The proportion of gold-standard substitutions that are included in the generated substitutions.
- **F1**: The harmonic mean between Precision and Recall.

Results The performance results obtained for all generators are presented in Table 8.10.

Generator	Potential	Precision	Recall	F1
Yamamoto	0.314	0.026	0.061	0.037
Merriam	0.473	0.046	0.106	0.065
Biran	0.414	0.084	0.079	0.081
Kauchak	0.506	0.078	0.103	0.089
WordNet	0.485	0.092	0.093	0.092
Glavas	0.661	0.105	0.141	0.121
Paetzold	0.791	**0.144**	0.198	**0.167**
All	**0.962**	0.043	**0.356**	0.076

Table 8.10 Benchmarking results for SG approaches

The results in Table 8.10 show that the method described in Chapter 5, which employs retrofitted context-aware word embedding models, yields the best Precision and F1 scores, and combining the output of all generation methods yields the highest Potential and Recall. The high Potential and Recall obtained by the combination of all generators shows that it would be the most appropriate strategy to compose an LS system in a scenario where the performance of both its Substitution Selection and Ranking approaches is perfect. The high Potential would allow the selector to find a simpler substitution for most complex words, and the high Recall would allow the ranker to select between a wider array of candidate substitutions, which should consequently result in simpler output.

Substitution Selection is, as discussed in Chapter 7, one of the most challenging tasks in the LS pipeline, which leads us to conclude that, in a realistic scenario, the most appropriate choice of Substitution Generation approach to compose an LS system would be the Paetzold

Generator, since its higher Precision would facilitate the selection process, and hence prevent ungrammatical and meaningless substitutions more effectively.

Substitution Selection

In this experiment, we benchmark the Substitution Selection systems in LEXenstein. As a baseline, we also include a "No Selection" strategy in the evaluation, which does not filter the candidate substitutions generated in any way. Since Substitution Selection requires a generated set of substitutions, we evaluate the performance of all combinations of SS and SG approaches.

Settings We configure the selectors of LEXenstein as follows:

- **First Sense**: Uses the same configuration described in Section 5.4.1.
- **Lesk Algorithm**: Uses the same configuration described in Section 5.4.1.
- **Path Similarity**: Uses the same configuration described in Section 5.4.1.
- **Belder Selector**: Uses the same configuration described in Section 5.4.1.
- **Biran Selector**: Uses the same configuration described in Section 5.4.1.
- **Unsupervised Boundary Ranker**: Uses the same features described in Section 7.2.2. The selector is trained over the candidates produced by all generators combined. We select seven candidates for each generator except the combination of all generators, for which we select 14 candidates. Hyper-parameters are maximised through 10-fold cross validation.

Datasets For a gold-standard, we use the NNSeval dataset.

Evaluation Metrics For evaluation, we use the same metrics described in the previous benchmarking of Substitution Generation approaches, which are Potential, Precision, Recall and F1.

Results The results obtained by all SS approaches are presented in Tables 8.11 through 8.18, respectively. The performance scores reported show that, while combining our Unsupervised Boundary Selector with all generators together produces the highest F1 values, combining the Belder Selector with the Devlin Generator yields the highest Precision. However, while the Belder Selector severely compromises the Potential and Recall of generators,

Selector	Potential	Precision	Recall	F1
First Sense	0.033	0.020	0.005	0.008
Lesk Algorithm	0.109	0.030	0.018	0.022
Path Similarity	0.013	0.012	0.002	0.004
Biran Selector	0.201	0.052	0.044	0.047
Belder Selector	0.126	**0.167**	0.018	0.033
Unsupervised Boundary Selector	0.427	0.118	0.088	**0.101**
No Selection	**0.473**	0.046	**0.106**	0.065

Table 8.11 Benchmarking results for SS approaches with respect to substitutions generated by the Merriam generator

Selector	Potential	Precision	Recall	F1
First Sense	0.004	0.009	0.001	0.001
Lesk Algorithm	0.021	0.021	0.003	0.006
Path Similarity	0.000	0.000	0.000	0.000
Biran Selector	0.146	0.028	0.027	0.027
Belder Selector	0.038	**0.205**	0.005	0.010
Unsupervised Boundary Selector	0.285	0.088	0.051	**0.065**
No Selection	**0.314**	0.026	**0.061**	0.037

Table 8.12 Benchmarking results for SS approaches with respect to substitutions generated by the Yamamoto generator

Selector	Potential	Precision	Recall	F1
First Sense	0.059	0.078	0.009	0.017
Lesk Algorithm	0.109	0.061	0.016	0.026
Path Similarity	0.008	0.042	0.002	0.003
Biran Selector	0.226	0.113	0.042	0.062
Belder Selector	0.130	**0.318**	0.019	0.036
Unsupervised Boundary Selector	0.473	0.144	0.089	**0.110**
No Selection	**0.485**	0.092	**0.093**	0.092

Table 8.13 Benchmarking results for SS approaches with respect to substitutions generated by the Devlin generator

Selector	Potential	Precision	Recall	F1
First Sense	0.042	0.078	0.006	0.010
Lesk Algorithm	0.084	0.090	0.012	0.022
Path Similarity	0.004	0.018	0.001	0.001
Biran Selector	0.226	0.109	0.039	0.057
Belder Selector	0.063	**0.283**	0.008	0.016
Unsupervised Boundary Selector	0.406	0.144	0.076	**0.099**
No Selection	**0.414**	0.084	**0.079**	0.081

Table 8.14 Benchmarking results for SS approaches with respect to substitutions generated by the Biran generator

Selector	Potential	Precision	Recall	F1
First Sense	0.025	0.019	0.003	0.006
Lesk Algorithm	0.050	0.033	0.008	0.013
Path Similarity	0.004	0.004	0.001	0.001
Biran Selector	0.251	0.103	0.049	0.067
Belder Selector	0.130	**0.276**	0.019	0.036
Unsupervised Boundary Selector	0.464	0.151	0.088	**0.117**
No Selection	**0.494**	0.134	**0.095**	0.106

Table 8.15 Benchmarking results for SS approaches with respect to substitutions generated by the Kauchak generator

Selector	Potential	Precision	Recall	F1
First Sense	0.042	0.027	0.006	0.010
Lesk Algorithm	0.151	0.049	0.023	0.032
Path Similarity	0.013	0.008	0.002	0.003
Biran Selector	0.289	0.114	0.059	0.078
Belder Selector	0.234	**0.225**	0.035	0.060
Unsupervised Boundary Selector	0.615	0.127	0.119	**0.123**
No Selection	**0.661**	0.105	**0.141**	0.121

Table 8.16 Benchmarking results for SS approaches with respect to substitutions generated by the Glavas generator

our Unsupervised Boundary Ranker does not. It has been shown to provide considerable increases in Precision and F1 for almost all generators at little cost of Recall, which, in the LS pipeline, would reduce the chances of a ranker placing an ungrammatical or incorrect substitution first.

Selector	Potential	Precision	Recall	F1
First Sense	0.054	0.043	0.008	0.013
Lesk Algorithm	0.176	0.060	0.026	0.037
Path Similarity	0.013	0.011	0.002	0.003
Biran Selector	0.322	0.122	0.068	0.087
Belder Selector	0.247	**0.201**	0.034	0.058
Unsupervised Boundary Selector	0.644	0.192	0.131	0.156
No Selection	**0.791**	0.144	**0.198**	**0.167**

Table 8.17 Benchmarking results for SS approaches with respect to substitutions generated by the Paetzold generator

Selector	Potential	Precision	Recall	F1
First Sense	0.075	0.015	0.013	0.014
Lesk Algorithm	0.285	0.028	0.050	0.036
Path Similarity	0.021	0.005	0.003	0.004
Biran Selector	0.502	0.051	0.161	0.077
Belder Selector	0.339	**0.183**	0.054	0.083
Unsupervised Boundary Selector	0.912	0.142	0.266	**0.185**
No Selection	**0.962**	0.043	**0.356**	0.076

Table 8.18 Benchmarking results for SS approaches with respect to substitutions generated by all generators combined

The remaining approaches have been shown less reliable. Although the Biran Selector was able to outperform the Precision of the No Selection method for almost all sets of generated substitutions, it also consistently caused very noticeable decreases in Potential and Recall in all scenarios. All WSD approaches performed very poorly, yielding very unsatisfactory performance regardless of the generator used.

Substitution Ranking

In this experiment we benchmark the Substitution Ranking approaches in LEXenstein. We also compare their performance with the baseline of SemEval 2012, as well as the former state-of-the-art approach for SR (Jauhar and Specia, 2012). The SemEval 2012 baseline ranks candidates according to their frequencies in the Google 1T corpus, composed of over 1

trillion words. The approach of Jauhar and Specia (2012) uses a linear weighted model over the rankings produced by various metrics to determine the ranking of substitution candidates.

Settings We configure the rankers of LEXenstein as follows:

- **Metric Ranker**: We train rankers over the following 18 distinct metrics:

    1. The candidate's length.
    2. The candidate's number of syllables.
    3. The candidate's number of senses in WordNet.
    4. The candidate's number of synonyms in WordNet.
    5. The candidate's number of hypernyms in WordNet.
    6. The candidate's number of hyponyms in WordNet.
    7. The candidate's 3-gram language model probability in Simple Wikipedia.
    8. The candidate's 3-gram language model probability in the Brown corpus.
    9. The candidate's 3-gram language model probability in the SUBTLEX corpus.
    10. The candidate's 3-gram language model probability in the SubIMDB corpus.
    11. The candidate's n-gram language model probability with two words to the left and right in Simple Wikipedia.
    12. The candidate's n-gram language model probability with two words to the left and right in the Brown corpus.
    13. The candidate's n-gram language model probability with two words to the left and right in the SUBTLEX corpus.
    14. The candidate's n-gram language model probability with two words to the left and right in the SubIMDB corpus.
    15. The candidate's Age of Acquisition, as determined by the features produced by the bootstrapping algorithm introduced in Section 8.2.
    16. The candidate's Familiarity, as determined by the features produced by the bootstrapping algorithm introduced in Section 8.2.
    17. The candidate's Concreteness, as determined by the features produced by the bootstrapping algorithm introduced in Section 8.2.
    18. The candidate's Imagery, as determined by the features produced by the bootstrapping algorithm introduced in Section 8.2.

- **Biran Ranker**: The language model of complex and simple data required are trained over the Wikipedia and Simple Wikipedia corpora Kauchak (2013), respectively.

- **Bott Ranker**: The language model of simple data required is trained over the Simple Wikipedia corpus.

- **Yamamoto Ranker**: The word co-occurrence model required is trained over the same 7 billion word corpus used in the training of the word embedding models used by the Glavas and Paetzold generators in Section 8.5.1. The language model of simple data required is trained over the Simple Wikipedia corpus (Kauchak, 2013).

- **Horn Ranker**: Uses the same features used by Horn et al. (2014), which include the translation probability between a candidate and the target word, and n-gram frequencies from Wikipedia and Simple Wikipedia. We learn the translation probability model from the complex-to-simple parallel corpus of Wikipedia and Simple Wikipedia sentences introduced by Kauchak (2013). Hyper-parameters are maximised through 10-fold cross validation.

- **Glavas Ranker**: Uses the same features used in Glavaš and Štajner (2015), which include n-gram frequencies, the cosine similarity between the target word and a candidate, as well as the average cosine similarity between the candidate and all content words in the target word's sentence. The word embeddings model required is the same one used by the Glavas Generator in the experiment of Section 8.5.1.

- **Boundary Ranker**: We use the same collocational and psycholinguistic features used in the extrinsic evaluation of Section 8.2, which consist on the language model probabilities of all n-grams with windows from zero to two tokens to the left and right of the target word, as well as Age of Acquisition and Word Familiarity. 3-gram language models are trained over the SubIMDB corpus with SRILM. Hyper-parameters are maximised through 10-fold cross validation.

Datasets The dataset selected for training and testing is the one used in the English Lexical Simplification task of SemEval 2012, which contain 300 and 1,710 instances, respectively. Each instance is composed of a sentence, a target complex word, and a series of candidate substitutions ranked in order of simplicity by non-native speakers of the English language. We chose the SemEval 2012 datasets in order for the ranking results obtained to be compared with the previous state-of-the-art approach for Substitution Ranking.

Evaluation Metrics For evaluation, we use the metrics described in Section 6.3.2, which are: TRank-at-1, TRank-at-2 and TRank-at-3.

Ranker	TRank-at-1	TRank-at-2	TRank-at-3
Word Length	0.474	0.685	0.818
Syllable Count	0.400	0.635	0.773
Senses	0.394	0.646	0.811
Synonyms	0.363	0.618	0.791
Hypernyms	0.338	0.594	0.773
Hyponyms	0.336	0.576	0.763
Probability: Simple Wiki	0.570	0.806	0.915
Probability: Brown	0.596	0.818	0.921
Probability: SUBTLEX	0.618	0.854	0.936
Probability: SubIMDB	0.630	0.850	0.934
N-gram: Simple Wiki	0.612	0.827	0.921
N-gram: Brown	0.602	0.813	0.906
N-gram: SUBTLEX	0.622	0.830	0.919
N-gram: SubIMDB	0.630	0.850	0.934
Age Of Acquisition	0.564	0.792	0.895
Familiarity	0.574	0.811	0.915
Concreteness	0.243	0.462	0.643
Imagery	0.296	0.529	0.701
Biran Ranker	0.513	0.731	0.856
Bott Ranker	0.574	0.806	0.913
Yamamoto Ranker	0.517	0.766	0.891
Horn Ranker	0.624	0.843	0.933
Glavas Ranker	0.632	0.848	0.934
Boundary Ranker	**0.657**	**0.860**	**0.939**
SemEval	0.602	-	-
Google1T	0.585	-	-

Table 8.19 Evaluation results for SR approaches

Results The performance scores obtained by all ranking approaches are presented in Table 8.19. The results show that the Boundary Ranker outperforms all other approaches, including the former state-of-the-art. Another interesting conclusion that can be drawn from the results says respect with the effectiveness of language models trained over spoken text corpora. The SubIMDB metric rankers have outperformed the former state-of-the-art by a considerable margin, showing that the language model probabilities extracted from SubIMDB are much more effective in capturing word simplicity than the ones from any other corpus.

Full Pipeline Evaluation

In this experiment, we evaluate the performance of all combinations of SG, SS and SR approaches in a practical scenario. With this experiment, we hope to be able to get a better understanding on how these strategies perform when combined, and hopefully determine what the most effective approach for LS is. In total, we evaluate the performance of $1,344$ distinct combinations of 8 generators, 7 selectors and 24 rankers.

Settings The SG, SS and SR approaches are all configured in the same way as they were in the experiments of Sections 8.5.1, 8.5.1 and 8.5.1.

Datasets As a gold-standard, we use the NNSeval dataset. Supervised rankers were trained over the training portion of the SemEval 2012 dataset.

Evaluation Metrics For evaluation, we use the metrics described in Section 8.3.7, which are:

- **Precision**: The proportion of instances in which the highest ranking substitution is either the target complex word itself or is in the gold-standard.
- **Accuracy**: The proportion of instances in which the highest ranking substitution is not the target complex word itself and is in the gold-standard.
- **Changed Proportion**: The proportion of instances in which the highest ranking substitution is not the target complex word itself.

Results The performance scores obtained for each ranker with respect to a set of substitutions produced by a substitution generator and selected by a substitution selector are shown in Tables C.1 through C.56, which are given in this thesis' appendix.

If one's goal is to create a simplifier that makes as little inappropriate replacements as possible, using the Belder Selector seems to be the most appropriate choice. As shown in Table 8.20, it allows rankers to obtain Precision scores much higher than they do when paired with other selectors. That does, however, comes at the cost of Accuracy.

In Table 8.21 we show the performance of the system which combines our SG, SS and SR approaches described in Chapters 5, Chapters 6 and 7. Its Accuracy score is one of the highest among the system combinations evaluated. Nonetheless, the highest Accuracy scores among all $1,344$ systems evaluated is obtained by a less sophisticated system. It combines our SG and SS strategies, but instead of using our supervised Boundary Ranker,

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.695	0.126	0.431
Syllable Count	0.724	0.142	0.418
Senses	0.699	0.134	0.435
Synonyms	0.674	0.138	0.464
Hypernyms	0.603	0.134	0.531
Hyponyms	0.682	0.126	0.444
Frequency: Simple Wiki	0.498	0.084	**0.586**
Frequency: Brown	0.674	0.121	0.448
Frequency: SUBTLEX	0.611	0.130	0.519
Frequency: SubIMDB	0.649	0.130	0.481
N-gram: Simple Wiki	0.749	0.142	0.393
N-gram: Brown	0.728	0.121	0.393
N-gram: SUBTLEX	0.703	0.130	0.427
N-gram: SubIMDB	0.736	0.121	0.385
Age of Acquisition	0.644	0.117	0.473
Familiarity	0.661	0.126	0.464
Concreteness	0.674	0.096	0.423
Imagery	0.636	0.117	0.481
Biran Ranker	0.703	**0.155**	0.452
Bott Ranker	0.715	0.134	0.418
Yamamoto Ranker	0.820	0.109	0.289
Horn Ranker	0.820	0.134	0.314
Glavas Ranker	**0.870**	0.105	0.234
Boundary Ranker	0.707	0.138	0.431

Table 8.20 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Belder Selector

which obtained the highest performance scores in our Substitution Ranking benchmarking, it ranks candidates according to single word frequencies from the SubIMDB corpus.

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.272	0.272	1.000
Syllable Count	0.218	0.218	1.000
Senses	0.255	0.255	1.000
Synonyms	0.218	0.218	1.000
Hypernyms	0.226	0.226	1.000
Hyponyms	0.205	0.205	1.000
Frequency: Simple Wiki	0.109	0.109	1.000
Frequency: Brown	0.280	0.280	1.000
Frequency: SUBTLEX	0.079	0.079	1.000
Frequency: SubIMDB	**0.310**	**0.310**	**1.000**
N-gram: Simple Wiki	0.285	0.285	1.000
N-gram: Brown	0.276	0.276	1.000
N-gram: SUBTLEX	0.285	0.285	1.000
N-gram: SubIMDB	0.272	0.272	1.000
Age of Acquisition	0.109	0.109	1.000
Familiarity	0.309	0.309	1.000
Concreteness	0.159	0.159	1.000
Imagery	0.247	0.247	1.000
Biran Ranker	0.272	0.272	1.000
Bott Ranker	0.276	0.276	1.000
Yamamoto Ranker	0.280	0.280	1.000
Horn Ranker	0.490	0.218	0.728
Glavas Ranker	0.485	0.238	0.753
Boundary Ranker	0.297	0.297	1.000

Table 8.21 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Unsupervised Boundary Ranking Selector

We can learn a lot from the results about the rankers themselves. The performance of the Horn ranker, which uses a very sophisticated supervised ranking strategy, varied a lot with respect to Precision and Accuracy scores. The Boundary Ranker, on the other hand, shows more consistent performance, specially when paired with generators that use word embedding models. This phenomenon suggests that the Boundary Ranking strategy suffers less from poor generalisation than the SVM Ranker, and is hence a more reliable approach for supervised Substitution Ranking.

But new questions arise along with these interesting findings. There is a clear inconsistency in our results: the winning system of the full pipeline evaluation is not composed by the combination of winning Substitution Generation, Selection and Ranking strategies from

the benchmarks for each step of the pipeline. Although it is expected that two evaluation methods could find different winners, in this case, inconsistencies such as this one makes it difficult to decide which system one should choose in order to obtain the most reliable simplifier. Should it be the combination of systems which obtained the highest performance in each step of the pipeline individually? Or should it be the winning system from the full pipeline evaluation? And how do the winning systems of other benchmarks fair against them?

To answer these questions, we conducted a human evaluation of our winning lexical simplifiers, as described in Section 8.5.2.

8.5.2 Human Evaluation

In this study, we attempt to find the most effective lexical simplifier in practice, given the needs of non-native English speakers. To do so, gathered non-native feedback on the quality of various simplifiers, each of which is the highest performing simplifier over a distinct evaluation strategy. By doing so, we are able not only to determine which simplifier non-native speakers of English prefer, but also to find evidence of which evaluation strategy is the most reliable.

Evaluation Strategies and Best Systems

We compared three evaluation strategies:

- **Pipeline**: Determines the best simplifier based on the performance of its approaches to Substitution Generation, Selection and Ranking individually. In other words, the simplifier is the combination of the best performing generator, selector and ranker, as determined by our pipeline step benchmarks in Sections 8.5.1, 8.5.1 and 8.5.1. According to our results, the best performing system under this evaluation strategy is composed by:

  - The generator described in Chapter 5, which uses retrofitted context-aware word embedding models.
  - The selector described in Chapter 7, which employs unsupervised Boundary Ranking.
  - The ranker described in Chapter 6, which employs supervised Boundary Ranking.

  We hereon address this system as the **Pipeline simplifier**.

- **NNSeval**: Determines the best simplifier based on the Accuracy obtained in a full pipeline evaluation conducted over the NNSeval dataset, described in the previous Section. According to our benchmarking results, the best performing system under this evaluation strategy is composed by:

  - The generator described in Chapter 5, which uses retrofitted context-aware word embedding models.
  - The selector described in Chapter 7, which employs unsupervised Boundary Ranking.
  - A simple frequency-based ranker, which ranks candidates according to their language model probability in the SubIMDB corpus.

  We hereon address this system as the **NNSeval simplifier**.

- **LexMTurk**: Determines the best simplifier based on the Accuracy obtained in a full pipeline evaluation conducted over the LexMTurk dataset. According to the results obtained by Glavaš and Štajner (2015), the best performing system under this evaluation strategy is composed by:

  - The Glavas generator, introduced by Glavaš and Štajner (2015), which uses typical word embedding models.
  - The Glavas ranker, introduced by Glavaš and Štajner (2015), which ranks candidates according to the average ranking obtained over various collocational and semantic features.

  Notice that this simplifier does not employ any explicit Substitution Selection strategy. We hereon address this system as the **LexMTurk simplifier**.

All simplifiers were configured in the same way described in the full pipeline evaluation of Section 8.5.1.

Methodology

We first randomly collected $1,200$ sentences from Wikipedia that contained at least one target content word (verb, noun, adjective or adverb) deemed complex by at least one non-native English speaker in the user study described in Section 3.1 with at least one synonym in WordNet, and which was not a color, proper name, number or stop-word. We did not consider synonyms which were either morphological variants of the target itself, or had a Levenshtein

distance (Levenshtein, 1966) to the target that was smaller than three characters. We then employed each of the previously mentioned systems in producing a simpler alternative to the target content word of each sentence. Notice that Wikipedia is the same source from which the sentences in LexMTurk and NNSeval were extracted.

For each sentence we created a "candidate pool", containing all three simpler alternatives produced by the systems (one produced by each system), as well as the target content word itself. One annotation instance was created for each sentence. Instances were composed by the candidate pool and a modified version of the sentence in which the target content word is replaced by a "gap" marker.

Through an online interface, annotators were asked to rank the candidates in a pool according to "*how well they fit the gap in the sentence*". A screenshot of the interface used is shown in Figure 8.1.

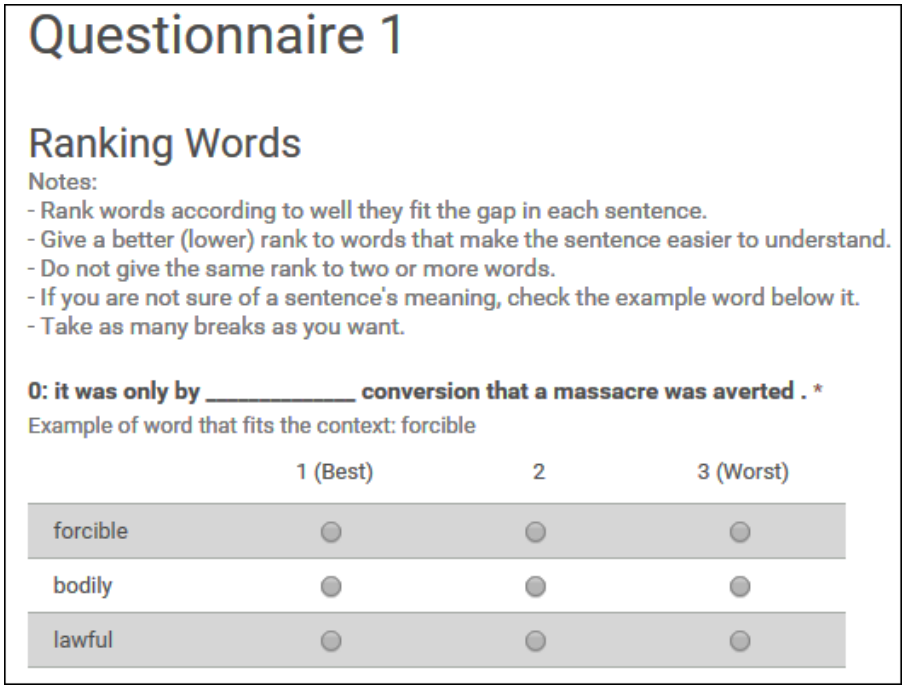

Fig. 8.1 Human evaluation interface

A total of 300 non-native English speakers participated, all university students and staff. Each annotator was presented with 20 instances, and each instance was annotated by 5 distinct annotators. Annotators were allowed to give the same ranking to two or more candidates.

We hypothesise that, when judging the "fitness" of candidates, non-native English speakers account for various contextual aspects, such as the sentences' grammaticality, meaning preservation and simplicity. This way, we simplify the annotation process, consequently

avoiding the costs inherent to training annotators to correctly judge these individual linguistic properties.

Results

We have found an average of 0.425 Spearman correlation between the five annotators' rankings, which reveals a considerable heterogeneity in their judgment. We must point out that despite the subjective nature of the task, annotators in this experiment have achieved a higher agreement than those who have produced the datasets for the English Lexical Simplification task of SemEval 2012 (0.398), which are also composed by rankings produced by non-native English speakers.

Table 8.22 shows the average ranking obtained for the simplifications produced by each system, as well as the target content word itself. It reveals substantial differences between the target word and the simplifications. The histogram in Figure 8.2 sheds light on this phenomenon: the target word is the one to best fit the gap in the sentence two times out of three, and rarely ever is at the bottom of the ranking.

System	Average
Pipeline	2.40 ± 0.85
NNSeval	2.49 ± 0.89
LexMTurk	2.50 ± 0.91
Target	$\mathbf{1.48 \pm 0.76}$

Table 8.22 Average rankings obtained by the candidates of each system evaluated

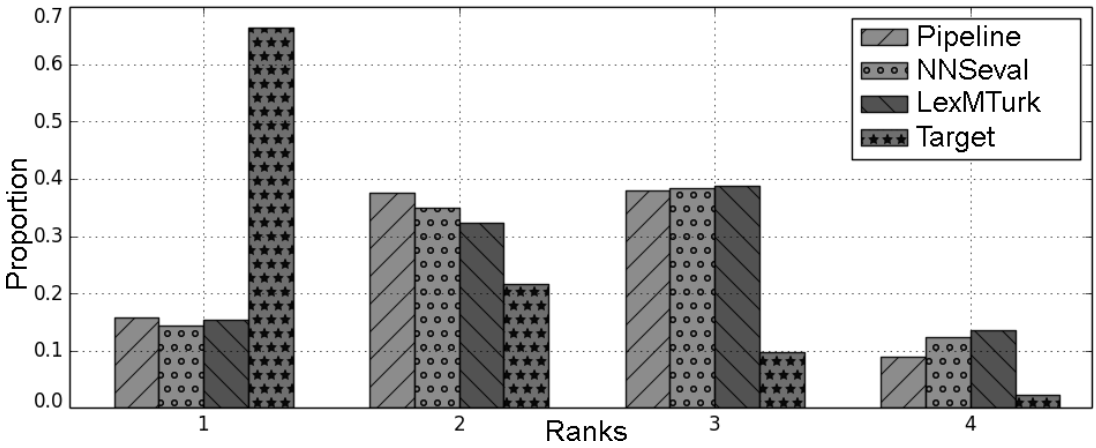

Fig. 8.2 Proportion over ranks per system

However, when it comes to the simplifiers themselves, one can notice a small, yet consistent preference from the annotators for the alternatives produced by the Pipeline

simplifier. Its average rank is 0.1 smaller than that of other simplifiers, and its variance is 0.05% smaller also. It was also ranked second an average of 3.92% times more frequently than the other simplifiers, and ranked last an average of 4.08% less. The F-test results depicted in Table 8.23, in which ♦ indicates $p < 0.05$ and ◊ otherwise, determine that this difference is statistically significant. F-tests were performed over the systems' ranking distributions.

In contrast, there is not a significant difference between the LexMTurk and NNSeval simplifiers, meaning that the quality of their simplifications is very similar. This phenomenon is rather surprising, since, as pointed out in Section 8.4, NNSeval was created with the intent of better capturing the needs non-native English speakers, and the target words selected for this experiment were deemed complex by non-native English speakers themselves.

	Pipeline	NNSeval	LexMTurk
NNSeval	♦		
LexMTurk	♦	◊	
Target	♦	♦	♦

Table 8.23 Results of F-tests performed over the distributions of rankings

These results raise, consequently, two important questions about the systems being evaluated:

1. Why do human evaluators prefer the simplifications produced by the Pipeline as opposed to the NNSeval simplifier?

2. If the NNSeval and LexMTurk simplifiers use such distinct strategies, then what makes their simplifications so similar?

We believe that the answer to the first question lies in the only difference that the Pipeline and NNSeval simplifiers have between them: the Substitution Ranking approach used. Both of them use the same generator and selector, but while the NNSeval simplifier employs a very simple frequency-based ranker, the Pipeline ranker uses a Boundary Ranker trained over manually annotated data.

Table 8.24 shows the percentage of simplification problems for which each pair of simplifiers have produced identical simplifications. Analysing the results, the first thing we were able to notice is that, due to the fact that they share most of their architecture, the simplifications produced by the Pipeline and NNSeval simplifiers are the same almost 60% of the time. These findings greatly contrast with the statistical significance tests depicted in Table 8.23: while we found no statistically significant difference ($p > 0.05$) between the

NNSeval and LexMTurk simplifiers, which have produced identical simplifications for barely more than 4% of instances, there is a statistically significant difference between the Pipeline and NNSeval simplifiers, of which the output is identical almost two out of three times.

	Pipeline	NNSeval	LexMTurk
Pipeline	100%		
NNSeval	59%	100%	
LexMTurk	3.5%	4.2%	100%

Table 8.24 Percentage of identical simplifications produced

By taking a closer look at the simplifications produced by the systems, the preference of non-native speakers of English for the Pipeline simplifier and the similarities between the NNSeval and LexMTurk simplifier become much more clear. Table 8.25 shows several examples of simplification problems from our experiment along with the simplifications produced by each system.

The first three examples represent a recurring phenomenon that highlights why the Pipeline simplifier is preferred by non-native English speakers. In all of them, the simplifications produced by the Pipeline simplifier are grammatical, and are the only one to resemble the meaning of the original target word in context. Consequently, in all of these examples, the Pipeline simplifications were ranked either first or second, while the NNSeval and LexMTurk ones were in third or fourth.

We have also noticed that the LexMTurk simplifier tends to produce spurious simplifications, such as the one in the second example, much more frequently than the other systems. Some other examples of ungrammatical and/or incoherent simplifications are “movement–the”, “www.manolocaracol.net” and “600-502-33”. We believe that the lack of a context-aware strategy for Substitution Generation and the absence of a Substitution Selection step are the main causes for this.

But more importantly yet, our qualitative analysis provides a better outline of the merits inherent to using a supervised Boundary Ranker as opposed to a simple frequency-based alternative. The three last examples of Table 8.25 represent another recurring phenomenon in our human evaluation: the NNSeval simplifier often fails to preserve the grammaticality of the sentence by replacing verbs, nouns and adjectives with morphologically incompatible alternatives. The error most frequently made by the NNSeval simplifier is replacing a verb with another in a different tense, such as it has done in the fourth and fifth examples.

In the following Section, we introduce an error analysis framework that allows a better outline of the strengths and weaknesses of the approaches evaluated in this benchmark.

It also reacts violently with water, spewing _____ liquid all over.

Target	NNSeval	LexMTurk	Pipeline
corrosive	bitter	dangerous	acid

A medical abortion is a type of non-surgical abortion in which abortifacient pharmaceutical drugs are used to induce _____.

Target	NNSeval	LexMTurk	Pipeline
abortion	pregnancy	controvserial	miscarriage

In other cases, they are used to model a more abstract process, and are the theoretical _____ of an algorithm.

Target	NNSeval	LexMTurk	Pipeline
underpinning	understanding	framework	basis

In 1978 “action office II” was renamed simply “action office”, and by 2005 had _____ sales totalling $5 billion.

Target	NNSeval	LexMTurk	Pipeline
attained	achieve	achieved	accomplished

The committee directs corporate management, approves major capital expenditures, establishes broad policy and monitors management’s performance in _____ the business and affairs of the corporation.

Target	NNSeval	LexMTurk	Pipeline
conducting	assist	monitoring	directing

Trajan’s wife and his friend licinius sura were _____ towards hadrian, and he may well have owed his succession to them.

Target	NNSeval	LexMTurk	Pipeline
well-disposed	kindly	good-nature	sympathetic

Table 8.25 Simplification examples produced by the systems evaluated

8.6 PLUMBErr: An Error Identification Framework

In order to assess and compare the performance of different LS strategies, previous work has used both manual and automatic evaluation methods. The most widely used is human evaluation. Biran et al. (2011), Paetzold and Specia (2013) and Glavaš and Štajner (2015) all use a very similar approach: they present human subjects with various simplifications produced by their systems and ask them to make judgments with respect to grammaticality, meaning preservation and simplicity, either individually or jointly. The methodology used in their work resembles that used in the human evaluation described in Section 8.5.2, in which we compare the performance of three LS approaches. Although human evaluation is a good way of assessing the effectiveness of an approach with respect to the needs of a target audience, it has limitations. Besides being a costly process that prevents one from performing large benchmarks, human evaluations tend to lead to inconclusive results due to the disagreement between judges. As discussed in Section 3.2, human annotators tend to have very unique definitions and standards with respect to complex linguistic properties such as grammaticality and meaning preservation. Yet another limitation of the human evaluation methodologies used in literature is the fact that they do not incorporate the step of Complex Word Identification. Because most authors choose to neglect CWI (Biran et al., 2011; Glavaš and Štajner, 2015; Horn et al., 2014; Paetzold and Specia, 2013), they randomly select the target words which are to be simplified by their systems and then judged by the subjects, which consequently ignores to the needs of the target audience being addressed.

Automatic evaluation approaches are much different. The most widely used method is the one introduced by Horn et al. (2014), in which the simplifications produced by a system for a set of problems are compared to a gold-standard produced by hundreds of humans through various metrics. But as demonstrated in the benchmarks of Section 8.5, even though this evaluation method is much less costly to perform, it is difficult to determine conclusively which of the systems compared best suits the needs of a target audience without the results from a human evaluation experiment, since distinct metrics and datasets tend to lead to different results. And as discussed in Section 8.5.1, typical full pipeline evaluation methods do not accommodate CWI, which is a crucial step in ensuring that a simplifier will fit the needs of a target audience.

Another limitation of typical human and automatic evaluation methods is the fact that they do not provide detailed insight on what are the strengths and limitations of the simplifier. Although a human evaluation in which subjects are asked to individually judge different aspects of a simplification can highlight the problem of a simplifier that makes frequent ungrammatical replacements, for example, it is not able to outline the reason why the simplifier does so.

Shardlow (2014a) introduces a solution to the aforementioned problems. Their approach uses human evaluation not to assess the quality of simplifications, but rather to verify the correctness of each decision made by a simplifier with respect to the usual pipeline. Although innovative, their error categorisation approach is subject to the same limitation of other human evaluation strategies: humans judgments are costly to acquire. Such costs make the evaluation much harder, and can oblige those with limited resources to restrict themselves to evaluating only one or two systems with the judgments of just a few subjects, which can compromise the reliability of the results.

In this Section, we introduce PLUMBErr: an error analysis framework for LS that provides an accessible automatic approach for error identification. In what follows, we discuss the approach of Shardlow (2014a) in more detail (Section 8.6.1), describe the resources and methods used in PLUMBerr (Section 8.6.2), and present various experiments (Sections 8.6.4 through 8.6.6).

8.6.1 Error Analysis in Lexical Simplification

Shardlow (2014a) describes a pioneer effort in LS evaluation. Their study introduces an error analysis methodology that allows one to outline in detail the intricacies of a simplifier. Taking the usual LS pipeline as a basis, they first outline all possible types of errors that a system can make when simplifying a target word in a sentence:

- **Type 1**: No error. The system did not make any mistakes while simplifying this target word.
- **Type 2A**: The system mistook a target complex word for simple.
- **Type 2B**: The system mistook a target simple word for complex.
- **Type 3A**: The system did not produce any candidate substitutions for the target word.
- **Type 3B**: The system did not produce any simpler candidate substitutions for the target word.
- **Type 4**: The system replaced the target word with a candidate that compromises the integrity of the sentence.
- **Type 5**: The system replaced the target word with a candidate that does not simplify the sentence.

Finally, they establish a methodology for error identification that uses human assessments to judge the output produced by the simplifier after each step of the pipeline. Their methodology, which is illustrated in Figure 8.3, is very sensible, and unlike previous human evaluation strategies, it incorporates the step of Complex Word Identification.

But as previously discussed, acquiring human judgments is often costly, which can consequently limit the amount of systems which could be compared in a benchmark. In their work (Shardlow, 2014a) they are only able to assess the performance of one simplifier based on the judgments made by one annotator. Although they do learn interesting things about which errors are more frequently made by a basic simplifier, their analysis does not allow strengths and weaknesses of various Complex Word Identification, Substitution Generation, Selection and Ranking to be compared. PLUMBErr offers a solution to this problem.

8.6.2 An Automatic Alternative

PLUMBErr is a framework for the automatic identification of errors made by pipelined Lexical Simplification systems. To produce a full report on the types of errors made by a lexical simplifier, PLUMBerr employs the same overall error categorisation methodology introduced by Shardlow (2014a), but in order to avoid the need for human judgments, it resorts to a set of pre-computed gold-standards and a list of complex words produced by non-native English speakers.

The LS system being evaluated by PLUMBErr is first required to solve a series of pre-determined simplification problems present in the BenchLS dataset (Section 8.6.2). Through the PLUMBErr workflow, the judgments and resources produced by the system after each step of the pipeline are then compared to the gold-standards present in BenchLS, as well as the set of complex words present in NNSVocab (Section 8.6.2), which then allow errors to be found and categorised.

BenchLS

BenchLS is a dataset composed of 929 instances. Each instance contains a sentence, a target word, and various gold replacements suggested by English speakers from the U.S with a variety of backgrounds. Although these replacements are not guaranteed to make the sentence simpler, they do ensure that the sentence's grammaticality and meaning are preserved.

The instances of BenchLS are automatically corrected versions of the instances in two other datasets from literature:

- **LexMTurk**: Composed of 500 instances with sentences extracted from Wikipedia. The target word of each instance was selected based on word alignments between

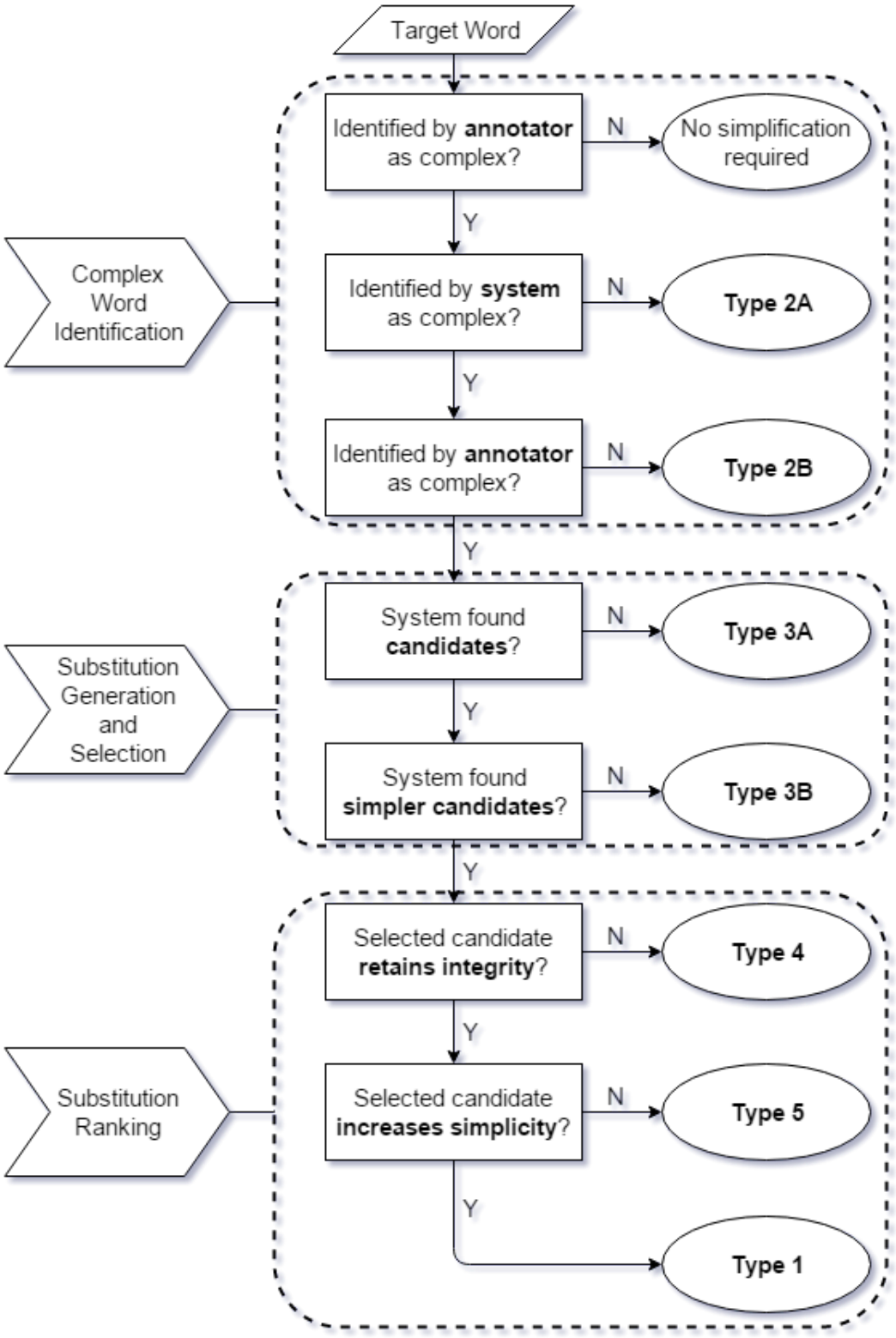

Fig. 8.3 Error categorisation methodology of Shardlow (2014a)

the sentence in Wikipedia and its equivalent simplified version in Simple Wikipedia. Candidate substitutions were produced by English speakers through the Amazon Mechanical Turk[8]. Each instance contains 50 candidate substitutions for the target word, each produced by a single annotator.

- **LSeval**: Composed of 439 instances with sentences extracted from the English Internet Corpus of English[9]. The target word of each instance was selected at random. Candidate substitutions were produced by English speakers through the Amazon Mechanical Turk, and then validated by PhD students.

The automatic correction steps used for BenchLS are two: spelling and inflection correction. For spelling, we use the algorithm introduced by Norvig[10] to fix any words with typos in them. For inflection, they resort to the Text Adorning module of LEXenstein to inflect any substitution candidates that are verbs, nouns, adjectives and adverbs to the same tense as the target word.

NNSVocab

NNSVocab is a vocabulary of $3,854$ words deemed complex by non-native English speakers. The words in NNSVocab were extracted from the data produced in the Complex Word Identification user study described in Section 3.1, in which 400 annotators judged $158,624$ words with respect to their complexity.

Each word in NNSVocab was deemed complex by at least one non-native English speaker in the user study, and is hence used as a representation of the simplification needs of this target audience.

Workflow

The workflow of PLUMBErr, which is illustrated in Figure 8.4, combines BenchLS and NNSVocab in a manner that allows for all error types described in Section 8.6.1 to be identified.

In its workflow, the system being evaluated first takes as input the target word from a simplification problem in BenchLS. The target word is then checked for complexity: is it in NNSVocab? i.e. has it been deemed complex by a non-native English speaker? If not, then it does not need to be simplified, otherwise, it does. The system then predicts the complexity of the word, which is again cross-checked in NNSVocab. If there is a disagreement between

[8]https://www.mturk.com
[9]http://corpus.leeds.ac.uk/internet.html
[10]http://norvig.com/spell-correct.html

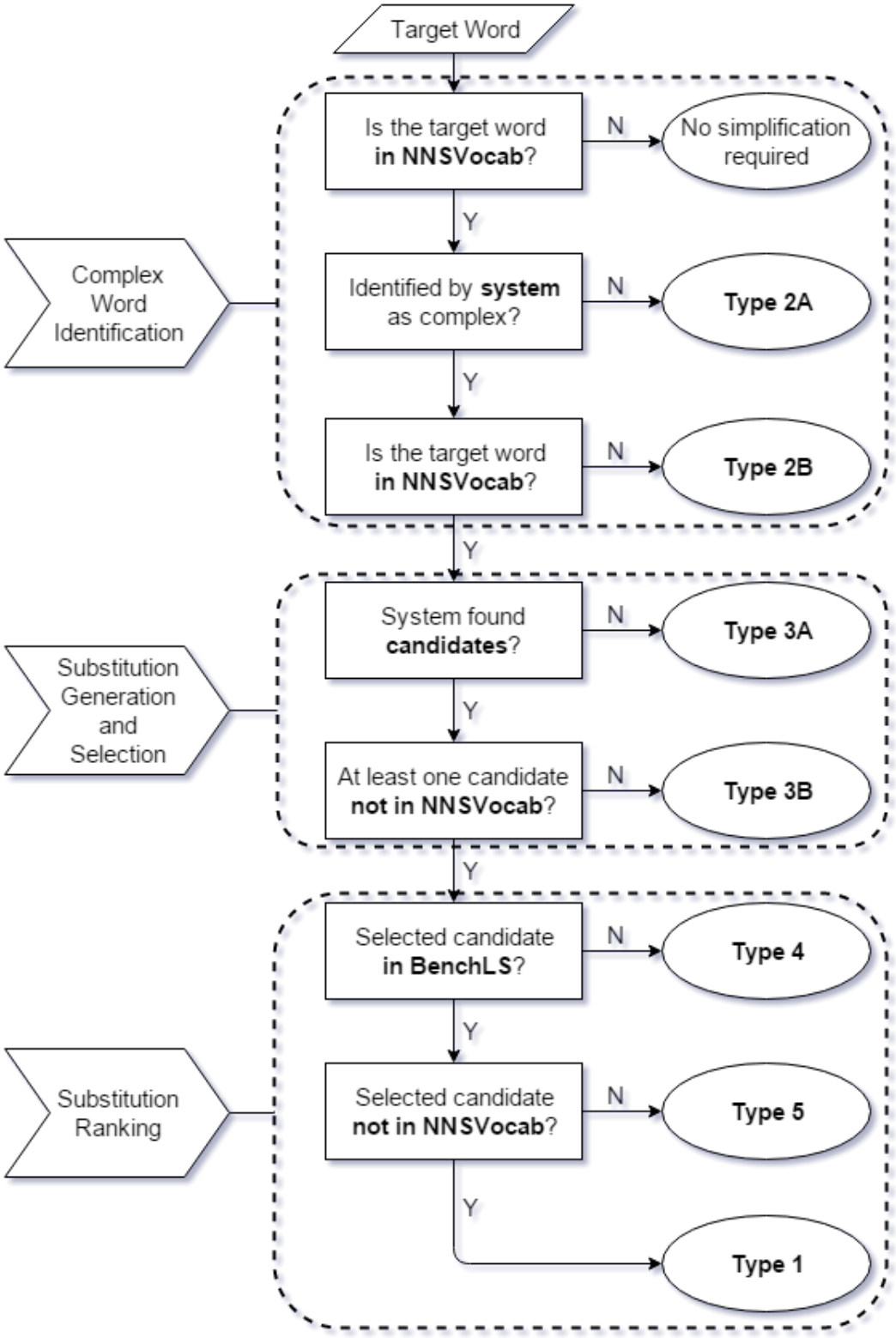

Fig. 8.4 The PLUMBErr methodology

the system's prediction and the judgment of non-native speakers of English, then an error of type 2 is identified. Otherwise, the system goes through the steps of Substitution Generation and Selection, and produces a set of candidate substitutions for the target complex word.

The candidates produced are then checked for errors of type 3. If there is at least one candidate available, and it is not a complex word in NNSVocab, then no errors are identified and the system moves on to ranking the candidates. After ranking, the best candidate among all is checked for errors of types 4 and 5: if the best candidate is among the replacements suggested by annotators in BenchLS, and it is not in NNSVocab, then it has successfully simplified the sentence.

At the end of the process, PLUMBErr produces a full report of the errors made in each of the problems present in BenchLS with respect to the needs of non-native English speakers represented in NNSVocab.

8.6.3 Experimental Settings

As previously mentioned, the work of Shardlow (2014a) features the error analysis of only one simplifier which does not perform any type form of Complex Word Identification or Substitution Selection. In order to showcase the potential of PLUMBErr, we have conducted an error categorisation benchmark with several LS systems with respect to all steps of the pipeline.

The systems chosen for our benchmark are four:

1. **The Devlin Simplifier**: The first lexical simplifier (Devlin and Tait, 1998). We choose it because it was featured in the error categorisation of Shardlow (2014a), and hence allows us to compare their manual error categorisation approach to our automatic strategy. Its approaches to each step of the pipeline are:

    - **Substitution Generation**: Extracts synonyms from WordNet.
    - **Substitution Selection**: Does not perform Substitution Selection.
    - **Substitution Ranking**: Uses Kucera-Francis coefficients.

2. **The Horn Simplifier**: We choose it because it is considered the current most effective supervised lexical simplifier, according to the experiments of Horn et al. (2014) and Glavaš and Štajner (2015). Its approaches to each step of the pipeline are:

    - **Substitution Generation**: Extracts complex-to-simple word correspondences from word alignments between Wikipedia and Simple Wikipedia.

- **Substitution Selection**: Does not perform Substitution Selection.
- **Substitution Ranking**: Learns a ranking model using Support Vector Machines from the examples in the LexMTurk dataset.

We use the same resources and parameters described in Section 8.5.1.

3. **The Glavas Simplifier**: We choose it because it is considered the current most effective unsupervised lexical simplifier, according to the experiments of Glavaš and Štajner (2015). Its approaches to each step of the pipeline are:

   - **Substitution Generation**: Extracts the 10 words closest to a given target complex word in a word embeddings model.
   - **Substitution Selection**: Does not perform Substitution Selection.
   - **Substitution Ranking**: Uses rank averaging over various semantic, lexical and collocational features.

   We use the same resources and parameters described in Section 8.5.1.

4. **The Paetzold Simplifier**: Our own simplifier, and the system that has performed best in the human evaluation experiment of Section 8.5.2. Its approaches to each step of the pipeline are:

   - **Substitution Generation**: Employs the generation strategy described in Chapter 5.
   - **Substitution Selection**: Employs the unsupervised Boundary Ranking strategy described in Chapter 7.
   - **Substitution Ranking**: Employs the supervised Boundary Ranking strategy described in Chapter 6.

   We use the same resources, features and configurations described in Section 8.5.1.

Notice that all aforementioned simplifiers have one thing in common: they do not employ a explicit Complex Word Identification step i.e. they simplify all words in a sentence. In order to make our experiments more meaningful and informative, we have decided to pair the lexical simplifiers selected with various CWI strategies:

1. **Simplify Everything (SE)**: Deems all words to be complex. We include this approach in our evaluation as a baseline.

2. **Support Vector Machines (SVM)**: Learns a word complexity model from training data using Support Vector Machines. As features, it uses the words' frequency and movie count in SUBTLEX, length, syllable, sense and synonym count. Syllables were obtained with the help of Morph Adorner. Sense and synonym counts were extracted from WordNet. We choose this approach because it is the first English language Complex Word Identification approach in literature that uses Machine Learning (Shardlow, 2013a).

3. **Threshold-Based (TB)**: Through brute force search, learns the threshold t from training data that best separates complex from simple words. For a metric, it uses raw word frequencies from Simple Wikipedia. We choose this strategy because it has achieved the highest F-score in the Complex Word Identification task of SemEval 2016, described in Section 3.1.5.

4. **Performance-Oriented Soft Voting (POSV)**: Combines several CWI strategies by weighting their predictions according to their overall performance in a validation dataset. This strategy is the one described in Section 3.1.5. We choose this approach because it has obtained the highest G-score (harmonic mean between Accuracy and Recall) in the Complex Word Identification task of SemEval 2016.

To train the supervised identifiers, we use the *optimistic* training set provided described in Section 3.1.4, which is the one used in the SemEval 2016 task. It contains $2,237$ instances, each composed of a target word in context and a binary label that receives value 1 if the word was judged complex by at least one non-native English speaker, or 0 otherwise.

8.6.4 Cumulative Analysis

In our cumulative analysis, we use the same error propagation technique of Shardlow (2014a), in which the errors made in a given step are carried onto to the next. If a simplifier makes a mistake in 90% of the instances during Complex Word Identification, for example, then it will only work over the 10% of instances left in the next pipeline step.

Table 8.26 shows raw count and proportion of instances in which errors of type 2A and 2B were made by each CWI approach. The fourth column of Table 8.26 features the sum of errors of type 2 (either 2A or 2B) made by each identifier. In analogous fashion, Table 8.27 shows the errors of type 3 made by each combination of CWI approach and simplifier. Table 8.28 shows the errors of type 4, 5 and 1 (no error) made by each combination.

The results in Table 8.26 reveal that Machine Learning approaches are much more reliable than the other two alternatives. Our Performance-Oriented System Voting method makes

	Error		
System	2A	2B	2
SE	0 (0%)	689 (74%)	689 (74%)
SVM	79 (9%)	268 (29%)	347 (37%)
TB	29 (3%)	645 (69%)	674 (73%)
POSV	104 (11%)	229 (25%)	**333 (36%)**

Table 8.26 Cumulative analysis results for errors of Type 2

the least amount of Type 2 errors. When it comes to the rest of the pipeline, Tables 8.27 and 8.28 show that the Paetzold simplifier is the clear winner. It performs the smallest total amount of Type 3 errors during Substitution Generation and Selection, and offers the smallest total proportion of Type 4 and 5 Substitution Ranking errors. It is also the most consistent: the errors of type 1 in Table 8.28 show that the Paetzold simplifier correctly simplifies the largest amount of instances across all CWI strategies. In practice, the combination between Performance-Oriented System Voting for CWI and the Paetzold simplifier yields the highest proportion of correctly simplified problems, which is in line with the previous observations.

		Error		
System		3A	3B	3
SE	Devlin	86 (36%)	34 (14%)	120 (50%)
SE	Horn	76 (32%)	43 (18%)	119 (50%)
SE	Glavas	70 (29%)	23 (10%)	93 (39%)
SE	Paetzold	**59 (25%)**	21 (9%)	**80 (33%)**
SVM	Devlin	140 (58%)	25 (10%)	165 (69%)
SVM	Horn	115 (48%)	30 (12%)	145 (60%)
SVM	Glavas	122 (51%)	17 (7%)	139 (58%)
SVM	Paetzold	120 (50%)	11 (5%)	131 (55%)
TB	Devlin	108 (45%)	23 (10%)	131 (55%)
TB	Horn	104 (43%)	36 (15%)	140 (58%)
TB	Glavas	91 (38%)	17 (7%)	108 (45%)
TB	Paetzold	80 (33%)	12 (5%)	92 (38%)
POSV	Devlin	165 (69%)	**7 (3%)**	172 (72%)
POSV	Horn	149 (62%)	17 (7%)	166 (69%)
POSV	Glavas	155 (65%)	8 (3%)	163 (68%)
POSV	Paetzold	149 (62%)	9 (4%)	158 (66%)

Table 8.27 Cumulative analysis results for errors of Type 3

System		Error		
		4	5	1
SE	Devlin	60 (50%)	17 (14%)	43 (36%)
SE	Horn	74 (61%)	15 (12%)	32 (26%)
SE	Glavas	81 (55%)	20 (14%)	46 (31%)
SE	Paetzold	68 (42%)	28 (18%)	64 (40%)
SVM	Devlin	36 (48%)	9 (12%)	30 (40%)
SVM	Horn	56 (59%)	13 (14%)	26 (27%)
SVM	Glavas	47 (47%)	17 (17%)	37 (37%)
SVM	Paetzold	41 (38%)	15 (14%)	53 (49%)
TB	Devlin	56 (51%)	15 (14%)	38 (35%)
TB	Horn	59 (59%)	11 (11%)	30 (30%)
TB	Glavas	68 (52%)	19 (14%)	45 (34%)
TB	Paetzold	60 (41%)	26 (18%)	62 (42%)
POSV	Devlin	36 (53%)	5 (7%)	27 (40%)
POSV	Horn	42 (57%)	10 (14%)	22 (30%)
POSV	Glavas	37 (48%)	**5 (6%)**	35 (45%)
POSV	Paetzold	**30 (37%)**	7 (9%)	**45 (55%)**

Table 8.28 Cumulative analysis results for errors of Type 4, 5 and 1 (no error)

8.6.5 Non-Cumulative Analysis

To complement the results from Section 8.6.4, we run a non-cumulative analysis of our simplifiers. In this analysis, the errors from the first pipeline step are not carried onto the next, i.e. it pairs each simplifier with a CWI system with 100% Accuracy. This setup allows us to evaluate the simplifiers in a scenario where the target words to be simplified are known, such as in the case of interactive user-driven LS approaches that allow for the user to select which words they find challenging (Azab et al., 2015; Devlin and Unthank, 2006; Rello et al., 2013a).

The results in Tables 8.29 and 8.30 emphasise the effectiveness of the LS approaches introduced in this thesis. Our generator and selector are the ones to make the smallest amount of Type 3 errors, and our ranker provides a noticeable 8% increase in the proportion of correctly simplified words (type 1 errors) over the second best simplifier.

8.6.6 Manual vs. Automatic

In order to trust the results obtained in our previous error analyses, we must assess the reliability of PLUMBErr. To do so, we compare our results with the ones reported by Shardlow (2014a), who also analyze the performance of the Devlin simplifier when paired

	Error		
System	3A	3B	3
Devlin	86 (36%)	34 (14%)	120 (50%)
Horn	76 (32%)	43 (18%)	119 (50%)
Glavas	70 (29%)	23 (10%)	93 (39%)
Paetzold	**59 (25%)**	**21 (9%)**	**80 (33%)**

Table 8.29 Non-cumulative analysis results for errors of Type 3

	Error		
System	4	5	1
Devlin	156 (65%)	41 (17%)	43 (18%)
Horn	176 (73%)	32 (13%)	32 (13%)
Glavas	164 (68%)	**30 (12%)**	46 (19%)
Paetzold	**137 (57%)**	39 (16%)	**64 (27%)**

Table 8.30 Non-cumulative analysis results for errors of Type 4, 5 and 1 (no error)

with a Simplify Everything identifier. The proportion of errors reported in the manual approach of Shardlow (2014a) and the automatic approach of PLUMBErr are reported in Figure 8.5.

While errors of Type 2 and 3 have very similar proportions, an interesting contrast was found for errors that occur during Substitution Ranking: the gold replacements present in BenchLS are more restrictive than the human judgments of Shardlow (2014a). Nonetheless, this phenomenon is expected, given that annotators of BenchLS were able to suggest only a single candidate substitution for each instance, without having access to other annotators' suggestions. This annotation approach compels them to suggest what they believe to be most appropriate replacement for the target word in question, consequently leading to a lot of repeated suggestions, and hence lower coverage.

8.7 Conclusions

In this Chapter, we presented various new resources and tools for LS, as well as a benchmark LS approaches and an error categorisation analysis of the best simplifiers available.

We introduced SubIMDB: a large structured corpus of subtitles of movies and series for the average audience, which aims at representing the everyday language to which non-native English speakers are accustomed. We found that word frequencies from SubIMDB capture lexical decision times more effectively than various other frequency norms. Additionally, we found that using only certain types of subtitles can yield noticeable increases in performance.

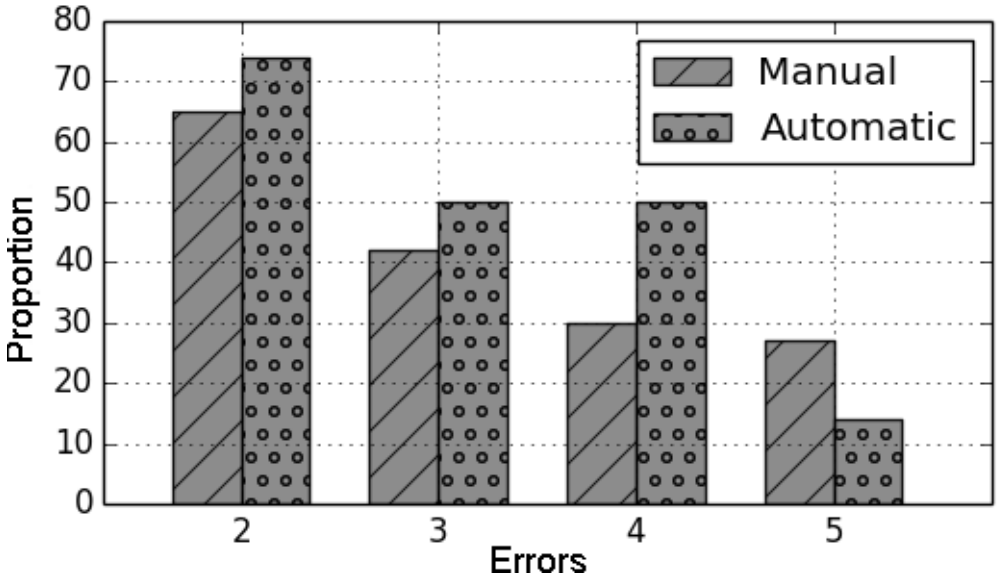

Fig. 8.5 Error proportion comparison

The same was observed for the prediction of other psycholinguistic properties, such as Familiarity, which, as demonstrated by Jauhar and Specia (2012), can be used in the creation of state-of-the-art LS strategies.

Exploiting word frequencies from SubIMDB as well as many other features, we created a bootstrapping algorithm for regression. It elaborates on early bootstrapping strategies for classification, and uses word embedding models in order to determine the confidence of a given prediction. We use our algorithm to train regressors that estimate four psycholinguistic properties included in the MRC Psycholinguistic Database: Familiarity, Age of Acquisition, Concreteness and Imagery. In our experiments, we compare our approach to various other baselines. The results reveal that our bootstrapped regressors are capable of estimating psycholinguistic properties much more effectively than all other approaches, offering upwards of 87% correlation with human-produced features.

In order to make our new resources and approaches available to the public, we developed LEXenstein, an open-source Python framework for LS. LEXenstein offers facilitated access not only to various approaches for all steps in the LS pipeline, but also to utilities often used in modern LS approaches, such as feature estimators, text adorners and spelling correctors.

By exploiting the great utility offered by LEXenstein and the aforementioned resources, we were able to present a performance comparison involving 8 Substitution Generators, 7 Substitution Selectors and 24 Substitution Rankers. As a gold-standard, we use NNSeval: a dataset for LS that accounts for the needs of non-native English speakers.

By evaluating each step in the LS pipeline individually, we found that, while extracting substitutions from the retrofitted context-aware word embeddings model introduced in Chapter 5 produces the least amount of spurious candidates, combining multiple generators allows for a much higher coverage, and could hence, in practice, increase the simplicity of the output produced by an LS approach. We have also found that, although Substitution Selection is a very challenging task, our unsupervised Boundary Ranking approach described in Chapter 7 is a more effective alternative to using standard classification and Word Sense Disambiguation strategies. In our comparison of Substitution Ranking systems, we found that our supervised Boundary Ranking approach described in Chapter 6 outperforms even modern ranking strategies, and is hence the current state-of-the-art approach for the task.

Our full pipeline evaluation, which compared $1,344$ distinct LS strategies, suggests that the current most effective approach for LS targeting non-native English speakers is entirely unsupervised, combining the SG strategy of Chapter 5, the unsupervised Boundary Ranker of Chapter 7 and single-word frequencies from the SubIMDB corpus. In our human evaluation, however, we have found that a full pipeline evaluation might not be the most reliable automatic evaluation approach for LS: it is more effective to create a simplifier by combining the highest performing approaches for each step of the pipeline individually.

In order to address the limitations of the automatic evaluation strategies used in our benchmark and provide with a more informative way of assessing the performance of lexical simplifiers, we introduced PLUMBErr: an automatic error categorisation framework for LS. PLUMBErr uses the same workflow for error identification proposed by Shardlow (2014a), but replaces on-demand human judgments with the data from two pre-annotated datasets: BenchLS and NNSVocab.

To showcase its utility, we used PLUMBErr to analyze the performance of 16 combinations of the most effective Complex Word Identification strategies and LS systems available. The results reveal that our Performance-Oriented Soft Voting identifier, which has obtained the highest G-score in the Complex Word Identification task of SemEval 2016, is the most reliable alternative among the ones evaluated. This observation supports the claim made in Section 3.1.4 that the G-score is a better indicator of an identifier's quality in practice than the traditional F-score. Our approaches to Substitution Generation, Selection and Ranking described in Chapters 5, 6 and 7 were also the ones to offer the most consistent results, obtaining the best overall performance for the remaining steps of the pipeline.

We released the SubIMDB corpus in both raw form, containing subtitles individually annotated with metadata, and in compiled form. Both versions are freely available for download at http://ghpaetzold.github.io/subimdb. The LEXenstein framework along with its documentation can be found at http://ghpaetzold.github.io/LEXenstein. The remaining

resources introduced in this Chapter, such as NNSeval and our bootstrapped psycholinguistic features can be downloaded at https://gustavopaetzold.wordpress.com/resources.

Chapter 9

Final Remarks

In this thesis, we presented our efforts towards understanding and addressing the Lexical Simplification needs of non-native English speakers. Through an extensive literature survey, we found that the intricacies of how non-native speakers of the English language communicate affect not only the effectiveness with which they can convey information in English, but also the way they are perceived by native speakers in academic settings. The literature review also revealed that, despite the fact that non-native English speakers number in millions, there have been barely any efforts in addressing their needs from a Lexical Simplification standpoint. The scarcity of such efforts led to a scarcity of tools and resources, which hampered progress in the area. We took this problem as the main motivation behind the contributions in this thesis.

We first conducted user studies with a total of $1,000$ non-native speakers of English. Words which pose challenges to them were found to be less ambiguous than simpler counterparts, and to occur less frequently in large corpora of text. No significant relationship between word complexity and its length or number of syllables was found. The number of words deemed complex by non-native speakers of English correlates with their level of proficiency, and seem to decrease with age. Our first results also revealed that automatically identifying complex words for non-native speakers of English is feasible. By combining the output of various Complex Word Identification strategies with a Performance-Oriented Soft Voting ensemble technique, we were able to create a strategy that captures 77% of complex words while still managing to correctly predict complexity 78% of the time. From our studies we also discovered that word embedding models provide useful insights on which alternatives can correctly replace complex words, and that the simplicity of such alternatives is, contrary to what early work claimed, strongly dependent on the context where they are found. Through these findings, we fulfilled two of our goals in this thesis, which are to

model and provide a better understanding of the simplification needs of non-native speakers of English.

Our survey allowed a clear outline of the state-of-the-art in Lexical Simplification to be drawn. Almost all previous work adheres to the traditional pipeline for the task. There are no simplifiers that attempt to jointly model all generation, selection and ranking processes inherent to Lexical Simplification. Using language models based on Recurrent Neural Networks, we were able to address this gap. Our Neural Language Models avoid the need for any annotated data or explicit generation, selection or ranking steps by simply learning to answer the question: which word from the vocabulary can best fill the gap left by a potentially complex word? Although promising in nature, our strategy was only able to perform competitively to other state-of-the-art simplifiers when incorporated as a step in the traditional Lexical Simplification pipeline. Following the success of previous work, we conceived an alternative pipelined Lexical Simplification approach. For Substitution Generation, we employ word embedding models . We bring novelty to the field by proposing retrofitted context-aware embedding models, which exploit both the complex words' grammatical form and their synonymy information to offer a state-of-the-art approach to Substitution Generation. To accompany our generator, we conceived a novel supervised Substitution Ranking strategy called Boundary Ranking, which learns how to rank candidate substitutions from a binary classification setup inferred from ranking examples that represent the needs of non-native English speakers. By exploiting n-gram language model probabilities from subtitles of movies and series for family and children, it becomes the state-of-the-art for the task. Its performance is further enhanced when psycholinguistic features, which we infer using a novel bootstrapping approach, are incorporated during training. To further improve our pipelined simplifier, we created a state-of-the-art Substitution Selection strategy. It treats Substitution Selection as a ranking problem, and exploits the counter-intuitive, and yet useful hypothesis that words are irreplaceable in order to train an unsupervised Boundary Ranker, which was shown to outperform all other selectors from previous work. With these contributions, we fulfill the third, fourth and fifth goals set for the completion of this thesis, which were to conceive novel, more effective approaches for each step of the Lexical Simplification pipeline.

Through our efforts in finding the most appropriate way to assess the effectiveness of simplifiers, we were able to offer new insight on which methodology should be used in Lexical Simplification evaluation. Using the information produced in our user studies, we introduce NNSeval: a new evaluation dataset that accounts for the needs of non-native speakers of English. NNSeval was created through the correction and filtering of two evaluation datasets introduced in past contributions. It allows for the components of pipelined simplifiers to be

assessed both individually and jointly. Using NNSeval and the datasets from the English Lexical Simplification task of SemEval 2012, we have conducted the largest benchmarking of Lexical Simplification systems to date. We compared the performance of 1,344 simplifiers in two ways: through evaluations for each individual step of the pipeline, and through a joint full pipeline evaluation of Substitution Generation, Selection and Ranking. Both methodologies suggest that our approaches to LS are the most effective available for non-native speakers of English. Nonetheless, our two evaluation methodologies have not led to the same winner, raising questions as to which one of them is the most reliable, and should hence be used in future work. We answer these questions through a human evaluation experiment, in which we highlight the importance of supervised Substitution Ranking strategies and discover that combining the winning approaches for each step of the pipeline individually is a more sensible option than adopting the winner of a full pipeline evaluation. Although our evaluation methodology may help one decide which simplifier to use for a given target audience, it does not accommodate the task of Complex Word Identification, nor does it allow for one to assess the simplifier's strengths and weaknesses in detail. Our automatic error analysis framework PLUMBErr offers a solution to this problem. By combining the data collected from our user studies with other consolidated LS datasets created in previous work, PLUMBErr automatically identifies and categorises errors made by pipelined simplifiers, and avoids any need of human annotation. In our experiments with PLUMBErr, we find more evidence of the effectiveness of our proposed approaches to the Lexical Simplification pipeline. Combining our Performance-Oriented Soft Voting approach to Complex Word Identification, retrofitted context-aware word embedding models for Substitution Generation, unsupervised Boundary Ranker for Substitution Selection, and supervised Boundary Ranker for Substitution Ranking, we create a complete approach that correctly identifies and simplifies complex words up to 25% more reliably than previous state-of-the-art simplifiers. These contributions fulfill the sixth and final established goal for this thesis, which was to find more informative and reliable evaluation strategies for Lexical Simplification.

In addition to these contributions, we have also created and released a wide array of new resources. The most significant among them is LEXenstein, a framework for Lexical Simplification. LEXenstein is a modular and easy-to-use Python library that provides dozens of approaches from previous work and our own to the various steps of the usual Lexical Simplification pipeline. It also provides functions commonly used by simplifiers, such as text adorning, spelling correction and others. Aditionally, we have contributed with a number of resources as a result of our user studies with non-native English speakers. The datasets comprise 211,564 human annotations, and are freely available. Using this data, we launched and organised the Complex Word Identification shared task of SemEval 2016,

in which 21 teams participated. Through this event, researchers from various institutions were able to familiarise with some of the challenges in Lexical Simplification and had the opportunity to present their approaches to Complex Word Identification. A total of 42 CWI systems were submitted, ranging from simple threshold-based to elaborate Machine Learning ensembles. The success of this shared task has granted more appeal and visibility to Lexical Simplification.

Despite our efforts and the positive outcome in their majority, our work reveals that Lexical Simplification, regardless of the target audience addressed, is still a challenging task. Modern simplifiers have not yet reached a point at which they can be reliably used to modify texts in practice, since their proficiency in capturing grammaticality and meaning preservation remain faint. Nevertheless, we believe our contributions will serve as a platform for future work which will be able to succeed in finding even more effective approaches to Lexical Simplification. In what follows, we suggest some possible directions towards this goal.

9.1 Future Work

- **New user studies**: There is no doubt that there is still much to be discovered about the needs of non-native English speakers from a Text Simplification standpoint. Our user studies highlight, in multiple occasions, the heterogeneity that characterises non-native speakers of English. We believe it would be very much revealing to conduct complementary user studies which focus strictly in the relationship between the difficulties faced by non-native English speakers and the intricacies of their native language. We hypothesise that the speakers of different native languages do have significantly distinct needs with respect to Lexical Simplification, and that a larger user study would bring light to these differences.

- **New shared tasks**: The Complex Word Identification task of SemEval 2016 was successful in bringing more visibility to Lexical Simplification and its challenges: the entry-level setup used for the task allowed for dozens of teams to participate. In the future, researchers should organise similar shared tasks that introduce other steps in the Lexical Simplification pipeline, such as Substitution Generation and Selection.

- **Incorporating phrases**: In this thesis, we focused on the challenges behind single-word replacements in Lexical Simplification. Nonetheless, it has long been established that, in numerous contexts, phrases are interchangeable with words, and can consequently be useful resources in Lexical Simplification. Certain types of phrases, such as

multi-word expressions, can also be deemed complex by the members of various target audiences. Future work should focus on the challenges inherent to phrase-to-phrase, phrase-to-word and word-to-phrase replacements.

- **Syntactic Simplification**: Syntactic Simplification can be helpful to those affected by conditions such as Aphasia, which can severely compromise one's capability of comprehending complex syntactic constructs. Although lexical and syntactic simplification are very distinct, it would be helpful to conduct analyses on the relationship between them. We hypothesise that Lexical Simplification can help Syntactic Simplification systems to more effectively prioritise certain portions of text, and hence avoid compromising its integrity. Searching for effective lexico-syntactic simplification strategies that joint model both tasks is a promising prospect to be explored.

References

Allauzen, C., Byrne, B., de Gispert, A., Iglesias, G., and Riley, M. (2014). Pushdown automata in statistical machine translation. *Computational Linguistics*, 40:687–723.

Aluisio, S. and Gasperin, C. (2010). *Proceedings of the NAACL 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, chapter Fostering Digital Inclusion and Accessibility: The PorSimples project for Simplification of Portuguese Texts, pages 46–53.

Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. (2012). Deep neural network language models. In *Proceedings of the NAACL 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28.

Austin, M., Paul, B., and Phil, B. (2006). Current state of english-language learners in the u.s. k-12 student population.

Azab, M., Hokamp, C., and Mihalcea, R. (2015). Using word semantics to assist english as a second language learners. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*.

Bacon, S. M. and Finnemann, M. D. (1990). A study of the attitudes, motives, and strategies of university foreign language students and their disposition to authentic oral and written input. *The Modern Language Journal*, 74(4):459–473.

Baddeley, A., Ellis, N., Miles, T., and Lewis, V. (1982). Developmental and acquired dyslexia: A comparison. *Cognition*, 11(2):185 – 199.

Baeza-Yates, R., Rello, L., and Dembowski, J. (2015). CASSA: A context-aware synonym simplification algorithm. In *Proceedings of the 2015 NAACL*, pages 1380–1385.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Balota, D. A. and Chumbley, J. I. (1984). Are lexical decisions a good measure of lexical access? the role of word frequency in the neglected decision stage. *Journal of Experimental Psychology: Human perception and performance*, 10:340–358.

Balota, D. A., Cortese, M. J., Sergent-Marshall, S. D., Spieler, D. H., and Yap, M. (2004). Visual word recognition of single-syllable words. *Journal of Experimental Psychology: General*, 133:283.

Balota, D. A., Yap, M. J., Hutchison, K. A., Cortese, M. J., Kessler, B., Loftis, B., Neely, J. H., Nelson, D. L., Simpson, G. B., and Treiman, R. (2007). The english lexicon project. *Behavior research methods*, 39:445–459.

Barbu, E., Martín-Valdivia, M. T., Martínez-Cámara, E., and Ureña-López, L. A. (2015). Language technologies applied to document simplification for helping autistic people. *Expert Systems with Applications*, 42:5076–5086.

Baron-Cohen, S. (2000). Theory of mind and autism: A review. *International review of research in mental retardation*, 23:169–184.

Basile, P., Caputo, A., and Semeraro, G. (2014). An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of the 25th COLING 2014*, pages 1591–1600.

Beck, D., Vlachos, A., Paetzold, G., and Specia, L. (2016). Shef-mime: Word-level quality estimation using imitation learning. In *Proceedings of the 1st WMT*, pages 772–776.

Begg, I. and Paivio, A. (1969). Concreteness and imagery in sentence meaning. *Journal of Verbal Learning and Verbal Behavior*, 8(6):821–827.

Bingel, J., Schluter, N., and Martínez Alonso, H. (2016). Coastalcph at semeval-2016 task 11: The importance of designing your neural networks right. In *Proceedings of the 10th SemEval*, pages 1028–1033.

Biran, O., Brody, S., and Elhadad, N. (2011). Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th ACL*, pages 496–501.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32.

Bott, S., Rello, L., Drndarevic, B., and Saggion, H. (2012). Can spanish be simpler? lexsis: Lexical simplification for spanish. In *Proceedings of 2012 COLING*, pages 357–374.

Bott, S. and Saggion, H. (2011). An unsupervised alignment algorithm for text simplification corpus construction. In *Proceedings of the 2011 MTTG*, pages 20–26.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992a). Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Brown, P. F., Pietra, V. J. D., Mercer, R. L., Pietra, S. A. D., and Lai, J. C. (1992b). An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18:31–40.

Brysbaert, M. and New, B. (2009a). Moving beyond kucera and francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–90.

Brysbaert, M. and New, B. (2009b). Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–990.

Burden, T. (2006). Second language acquisition: A new look at the implications of krashen's hypotheses. *Journal of Regional Development Studies*.

Burgess, C. and Livesay, K. (1998). The effect of corpus size in predicting reaction time in a basic word recognition task: Moving on from kučera and Francis. *Behavior Research Methods, Instruments, & Computers*, 30:272–277.

Burns, P. R. (2013). Morphadorner v2: A java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL.*

Campbell, N. (1987). Adapted literary texts and the efl reading programme. *ELT Journal*, 41(2):132–135.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22:249–254.

Carroll, J., Minnen, G., Canning, Y., Devlin, S., and Tait, J. (1998). Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Carroll, J., Minnen, G., Pearce, D., Canning, Y., Devlin, S., and Tait, J. (1999). Simplifying text for language-impaired readers. In *Proceedings of the 9th EACL*, pages 269–270.

Carroll, J. B. and White, M. N. (1973). Word frequency and age of acquisition as determiners of picture-naming latency. *The Quarterly Journal of Experimental Psychology*, 25(1):85–95.

Carver, R. P. (1994). Percentage of unknown vocabulary words in text as a function of the relative difficulty of the text: Implications for instruction. *Journal of Literacy Research*, 26(4):413–437.

Chandrasekar, R., Doran, C., and Srinivas, B. (1996). Motivations and methods for text simplification. In *Proceedings of the 16th COLING*, pages 1041–1044.

Chen, H.-B., Huang, H.-H., Chen, H.-H., and Tan, C.-T. (2012). A simplification-translation-restoration framework for cross-domain smt applications. In *Proceedings of 2012 COLING*, pages 545–560.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, pages 263–270. Association for Computational Linguistics.

Choubey, P. and Pateria, S. (2016). Garuda & bhasha at semeval-2016 task 11: Complex word identification using aggregated learning models. In *Proceedings of the 10th SemEval*, pages 1006–1010.

Cohn, T. and Lapata, M. (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674.

Coltheart, M. (1981). The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Coltheart, M. (1996). *Phonological Dyslexia*. Cognitive Neuropsychology. Psychology Press.

Connine, C. M., Mullennix, J., Shernoff, E., and Yelen, J. (1990). Word familiarity and frequency in visual and auditory word recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(6):1084.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Davoodi, E. and Kosseim, L. (2016). Clac at semeval-2016 task 11: Exploring linguistic and psycho-linguistic features for complex word identification. In *Proceedings of the 10th SemEval*, pages 982–985.

De Belder, J. and Moens, M.-F. (2010). Text simplification for children. In *Proceedings of the SIGIR Workshop on Accessible Search Systems*, pages 19–26.

De Belder, J. and Moens, M.-F. (2012a). A dataset for the evaluation of lexical simplification. In *Computational Linguistics and Intelligent Text Processing*, pages 426–437. Springer.

De Belder, J. and Moens, M.-F. (2012b). A dataset for the evaluation of lexical simplification. In *Computational Linguistics and Intelligent Text Processing*, pages 426–437.

Deléger, L. and Zweigenbaum, P. (2009). Extracting lay paraphrases of specialized expressions from monolingual comparable medical corpora. In *Proceedings of the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora*, pages 2–10.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Devinsky, O. and D'Esposito, M. (2003). *Neurology of Cognitive and Behavioral Disorders*.

Devlin, S. (1999). *Simplifying Natural Language for Aphasic Readers*. PhD thesis, University of Sunderland.

Devlin, S. and Tait, J. (1998). The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173.

Devlin, S. and Unthank, G. (2006). Helping aphasic people process online information. In *Proceedings of the 8th SIGACCESS*, pages 225–226.

Drndarević, B. and Saggion, H. (2012). Towards automatic lexical simplification in spanish: an empirical study. In *Proceedings of the 1st PITR*, pages 8–16.

Elhadad, N. (2006). Comprehending technical texts: Predicting and defining unfamiliarterms. In *Proceedings of the 2006 AMIA*.

Elhadad, N. and Sutaria, K. (2007). Mining a lexicon of technical terms and lay equivalents. pages 49–56.

Ellis, A. (1993). *Reading, Writing and Dyslexia: A Cognitive Analysis*. Open University Press.

Ellis, R. (1994). *The Study of Second Language Acquisition*. Oxford University Press.

Evert, S. (2010). Google web 1t 5-grams made easy (but not for the computer). In *Proceedings of the 2010 NAACL*, pages 32–40.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 NAACL*, pages 1606–1615.

Feblowitz, D. and Kauchak, D. (2013). Sentence simplification as tree transduction. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 1–10.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Francis, W. N. and Kucera, H. (1979). Brown corpus manual. *Brown University*.

Freyhoff, G., Hess, G., Kerr, L., Tronbacke, B., and Van Der Veken, K. (1998). Make it simple: European guidelines for the production of easy-to-read information for people with learning disability for authors, editors, information providers, translators and other interested persons. *ILSMH European Association*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Friedman, R. B. and Hadley, J. A. (1992). Letter-by-letter surface alexia. *Cognitive Neuropsychology*, 9(3):185–208.

Galaburda, A. M. (2006). Dyslexia: advances in cross-level research. *The Dyslexic Brain: New Pathways in Neuroscience Discovery*.

Gardner, D. and Hansen, E. C. (2007). Effects of lexical simplification during unaided reading of english informational texts. *TESL Reporter*, 40(2):27–59.

Gass, S. and Selinker, L. (2008). *Second Language Acquisition: An Introductory Course*. Topics in Applied Psycholinguistics Series. Taylor & Francis.

Gilhooly, K. J. and Logie, R. H. (1980). Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

Gitterman, M., Goral, M., and Obler, L. (2012). *Aspects of Multilingual Aphasia*. Communication Disorders Across Languages. Channel View Publications.

Glavaš, G. and Štajner, S. (2015). Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd ACL*, pages 63–69.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th ICML*, pages 513–520.

Goller, C. and Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the 1996 IEEE International Conference on Neural Networks*, pages 347–352.

Goodglass, H., Kaplan, E., and Barresi, B. (2001). *Assesment of Aphasia and Related Disorders 3rd - Boston Manual*. 3 edition.

Hearst, M. A. (1994). Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd ACL*, pages 9–16.

Hill, F. and Korhonen, A. (2014). Concreteness and subjectivity as dimensions of lexical meaning. In *Proceedings of ACL*, pages 725–731.

Hirsh, D., Nation, P., et al. (1992). What vocabulary size is needed to read unsimplified texts for pleasure? *Reading in a Foreign Language*, 8:689–689.

Honeyfield, J. (1977). Simplification. *TESOL Quarterly*, pages 431–440.

Horn, C., Manduca, C., and Kauchak, D. (2014). Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd ACL*, pages 458–463.

Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101.

Huebler, F. and Lu, W. (2012). Adult and youth literacy, 1990-2015: Analysis of data for 41 selected countries. *Education Indicators and Data Analysis*.

Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., and Daumé III, H. (2014). A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 EMNLP*, pages 633–644.

Janczura, G. A., Castilho, G. M. d., Rocha, N. O., van Erven, T. d. J. C., and Huang, T. P. (2007). Normas de concretude para 909 palavras da lingua portuguesa. *Psicologia: Teoria e Pesquisa*, 23:195 – 204.

Jauhar, S. and Specia, L. (2012). Uow-shef: Simplex–lexical simplicity ranking based on contextual and psycholinguistic features. In *Proceedings of the 1st SemEval*, pages 477–481.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM*, pages 133–142.

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the 12th SIGKDD*, pages 217–226.

Kajiwara, T., Matsumoto, H., and Yamamoto, K. (2013). Selecting proper lexical paraphrase for children. pages 59–73.

Kandula, S., Curtis, D., and Zeng-Treitler, Q. (2010). A semantic and syntactic text simplification tool for health content. In *Proceedings of the 2010 AMIA*, pages 366–70.

Kauchak, D. (2013). Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546.

Kauchak, D. (2016). Pomona at semeval-2016 task 11: Predicting word complexity based on corpus frequency. In *Proceedings of the 10th SemEval*, pages 1047–1051.

Kauchak, D. and Barzilay, R. (2006). Paraphrasing for automatic evaluation. In *Proceedings of the 2006 NAACL*, pages 455–462.

Keskisärkkä, R. (2012). Automatic text simplification via synonym replacement. Master's thesis, Linköping University.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*, pages 423–430.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 NAACL*, pages 48–54.

Konkol, M. (2016). Uwb at semeval-2016 task 11: Exploring features for complex word identification. In *Proceedings of the 10th SemEval*, pages 1038–1041.

Koo, T., Carreras Pérez, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of the 46th ACL*, pages 595–603.

Krasher, S. D. (1985). *The input hypothesis: issues and implications*. New York Longman.

Kuru, O. (2016). Ai-ku at semeval-2016 task 11: Word embeddings and substring features for complex word identification. In *Proceedings of the 10th SemEval*, pages 1042–1046.

Larsen-Freeman, D. (1991). Second language acquisition research: Staking out the territory. *Tesol Quarterly*, 25(2):315–350.

Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An Electronic Lexical Database*, 49(2):265–283.

Leki, I. (2001). "a narrow thinking system": Nonnative-english-speaking students in group projects across the curriculum. *TESOL quarterly*, 35(1):39–67.

Leow, R. P. (1993). To simplify or not to simplify. *Studies in Second Language Acquisition*, 15(03):333–355.

Leroy, G., Endicott, J. E., Kauchak, D., Mouradi, O., and Just, M. (2013). User evaluation of the effects of a text simplification algorithm using term familiarity on perception, understanding, learning, and information retention. *Journal of Medical Internet Research*, 15.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Conference on Systems Documentation*, pages 24–26.

Lesser, R. and Milroy, L. (1993). *Linguistics and Aphasia: Psycholinguistic and Pragmatic Aspects of Intervention*. Language in social life series. Longman.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Lewis, M. P., Simons, G. F., and Fennig, C. D. (2016). *Ethnologue: Languages of the world*. SIL international.

Ligozat, A.-L., Garcia-Fernandez, A., Grouin, C., and Bernhard, D. (2012). Annlor: a naïve notation-system for lexical outputs ranking. pages 487–492.

Lison, P. and Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th LREC*.

Lomax, R. (2007). *Statistical Concepts: A Second Course*. Lawrence Erlbaum Associates.

Long, M. H. and Ross, S. (1993). Modifications that preserve language and content.

Lotherington-Woloszyn, H. (1992). Simplification: The questionable role of interventionist language in esl reading comprehension. *Comprehension-Based Second Language Teaching*, pages 451–476.

Malmasi, S., Dras, M., and Zampieri, M. (2016). Ltg at semeval-2016 task 11: Complex word identification with classifier ensembles. In *Proceedings of the 10th SemEval*, pages 996–1000.

Malmasi, S. and Zampieri, M. (2016). Maza at semeval-2016 task 11: Detecting lexical complexity using a decision stump meta-classifier. In *Proceedings of the 10th SemEval*, pages 991–995.

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 EMNLP*, pages 133–139.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Martínez Martínez, J. M. and Tan, L. (2016). Usaar at semeval-2016 task 11: Complex word identification with sense entropy and sentence perplexity. In *Proceedings of the 10th SemEval*, pages 958–962.

Maziero, E. G., Pardo, T. A. S., Di Felippo, A., and Dias-da Silva, B. C. (2008). A base de dados lexical e a interface web do tep 2.0: Thesaurus eletronico para o portugues do brasil. In *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*, pages 390–392.

McCarthy, D. and Navigli, R. (2007). Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th SemEval*, pages 48–53.

Mihalcea, R., Sinha, R., and McCarthy, D. (2010). Semeval-2010 task 2: Cross-lingual lexical substitution. In *Proceedings of the 5th SemEval*, pages 9–14.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech*, 2.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of 2013 NAACL*, pages 746–751.

Milberg, W., Blumstein, S. E., and Dworetzky, B. (1987). Processing of lexical ambiguities in aphasia. *Brain and Language*, 31(1):138–150.

Moore, R. C. and DeNero, J. (2011). L1 and l2 regularization for multiclass hinge loss models. In *Proceedings of the 1st MLSLP*, pages 1–5.

Morrel-Samuels, P. and Krauss, R. M. (1992). Word familiarity predicts temporal asynchrony of hand gestures and speech. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(3):615.

Mukherjee, N., Patra, B. G., Das, D., and Bandyopadhyay, S. (2016). Ju_nlp at semeval-2016 task 11: Identifying complex words in a sentence. In *Proceedings of the 10th SemEval*, pages 986–990.

Nat, G. (2016). Sensible at semeval-2016 task 11: Neural nonsense mangled in ensemble mess. In *Proceedings of the 10th SemEval*, pages 963–968.

Nation, I. S. P. (2001). *Learning vocabulary in another language*. Ernst Klett Sprachen.

Nation, P. and Coady, J. (1988). Vocabulary and reading. *Vocabulary and language teaching*, 97:110.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys*, 41:10.

Navigli, R. and Ponzetto, S. P. (2010). Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th ACL*, pages 216–225.

Nunes, B. P., Kawase, R., Siehndel, P., Casanova, M. a., and Dietze, S. (2013). As simple as it gets - a sentence simplifier for different learning levels and contexts. In *Proceedings of the 13th ICALT*, pages 128–132.

Ogden, C. K. (1968). *Basic English: international second language*. Harcourt, Brace & World.

Oh, S.-Y. (2001). Two types of input modification and efl reading comprehension: Simplification versus elaboration. *TESOL quarterly*, 35(1):69–96.

Okazaki, N. (2007). CRFsuite: a fast implementation of Conditional Random Fields. http://www.chokkan.org/software/crfsuite/.

Ong, E., Damay, J., Lojico, G., Lu, K., and Tarantan, D. (2007). Simplifying text in medical literature. volume 4, pages 37–47.

Padró, L. and Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. In *Proceedings of the 2012 LREC*.

Paetzold, G. and Specia, L. (2016a). Simplenets: Machine translation quality estimation with resource-light neural networks. In *Proceedings of the 1st WMT*, pages 812–818.

Paetzold, G. H. (2013). *Um sistema de simplificação automática de textos escritos em inglês por meio de transduçao de árvores*. State University of Western Paraná.

Paetzold, G. H. (2015a). Morph adorner toolkit: Morph adorner made simple. http://ghpaetzold.github.io/MorphAdornerToolkit/.

Paetzold, G. H. (2015b). Reliable lexical simplification for non-native speakers. In *Proceedings of the 2015 NAACL Student Research Workshop*, pages 9–16.

Paetzold, G. H. (2015c). Using positional suffix trees to perform agile tree kernel calculation. In *Proceedings of the Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 269–275.

Paetzold, G. H. and Schemberger, E. E. (2015). Extlex: Um analisador léxico extensível capaz de detectar erros lexicais. *Revista Eletrônica Científica Inovação e Tecnologia*, 2:26–36.

Paetzold, G. H. and Specia, L. (2013). Text simplification as tree transduction. In *Proceedings of the 9th STIL*, pages 116–125.

Paetzold, G. H. and Specia, L. (2015). Lexenstein: A framework for lexical simplification. In *Proceedings of The 53rd ACL*, pages 85–90.

Paetzold, G. H. and Specia, L. (2016b). Benchmarking lexical simplification systems. In *Proceedings of the 10th LREC*.

Paetzold, G. H. and Specia, L. (2016c). Inferring psycholinguistic properties of words. In *Proceedings of the 2016 NAACL*, pages 435–440.

Paetzold, G. H. and Specia, L. (2016d). Multi-level quality prediction with quest++. In *Proceedings of the 19th EAMT*, page 379.

Paetzold, G. H. and Specia, L. (2016e). Plumberr: An automatic error identification framework for lexical simplification. In *Proceedings of the 1st QATS*.

Paetzold, G. H. and Specia, L. (2016f). Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th SemEval*, pages 560–569.

Paetzold, G. H. and Specia, L. (2016g). Simplenets: Evaluating simplifiers with resource-light neural networks. In *Proceedings of the 1st QATS*.

Paetzold, G. H. and Specia, L. (2016h). Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th SemEval*, pages 969–974.

Paetzold, G. H. and Specia, L. (2016i). Unsupervised lexical simplification for non-native speakers. In *Proceedings of The 30th AAAI*, pages 3761–3767.

Paetzold, G. H., Specia, L., and Savourel, Y. (2015). Okapi+quest: Translation quality estimation within okapi. In *Proceedings of the 18th EAMT*, page 222.

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of 2010 LREC*, pages 1320–1326.

Palakurthi, A. and Mamidi, R. (2016). Iiit at semeval-2016 task 11: Complex word identification using nearest centroid classification. In *Proceedings of the 10th SemEval*, pages 1017–1021.

Parker, K. and Chaudron, C. (1987). The effects of linguistic simplifications and elaborative modifications on l2 comprehension. *University of Hawai'i Working Papers in ESL*, 6(2):107–133.

Patterson, K., Marshall, J., and Coltheart, M. (1985). *Surface dyslexia: neuropsychological and cognitive studies of phonological reading*. Lawrence Erlbaum Associates.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pica, T. (1994). Questions from the language classroom: Research perspectives. *Tesol Quarterly*, 28(1):49–79.

Prather, P., Zurif, E., Love, T., and Brownell, H. (1997). Speed of lexical activation in nonfluent broca's aphasia and fluent wernicke's aphasia. *Brain and Language*, 411(59):391–411.

Prather, P., Zurif, E., Stern, C., and Rosen, T. J. (1992). Slowed lexical access in nonfluent aphasia: A case study. *Brain and Language*, 43(2):336–348.

Quijada, M. and Medero, J. (2016). Hmc at semeval-2016 task 11: Identifying complex words using depth-limited decision trees. In *Proceedings of the 10th SemEval*, pages 1034–1037.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the 13th CONLL*, pages 147–155.

Rayner, K. and Duffy, S. A. (1986). Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & Cognition*, 14:191–201.

Reiter, E., Dale, R., and Feng, Z. (2000). *Building natural language generation systems*, volume 33. MIT Press.

Rello, L., Baeza-Yates, R., Bott, S., and Saggion, H. (2013a). Simplify or help?: text simplification strategies for people with dyslexia. In *Proceedings of the 10th W4A*, pages 1–10.

Rello, L., Baeza-Yates, R., Dempere-Marco, L., and Saggion, H. (2013b). Frequent words improve readability and short words improve understandability for people with dyslexia. *Human-Computer Interaction*, pages 203–219.

Rello, L., Bautista, S., Baeza-Yates, R., Gervás, P., Hervás, R., and Saggion, H. (2013c). One half or 50%? an eye-tracking study of number representation readability. *Human-Computer Interaction*, pages 229–245.

Reves, T. and Medgyes, P. (1994). The non-native english speaking EFL/ESL teacher's self-image: An international survey. *System*, 22(3):353–367.

Robbins, T. (2003). *Jitterbug Perfume*. Random House Publishing Group.

Rong, X. (2014). word2vec parameter learning explained. *CoRR*, abs/1411.2738.

Ronzano, F., Abura'ed, A., Espinosa Anke, L., and Saggion, H. (2016). Taln at semeval-2016 task 11: Modelling complex words by contextual, lexical and semantic features. In *Proceedings of the 10th SemEval*, pages 1011–1016.

Rudell, A. P. (1993). Frequency of word usage and perceived word difficulty: Ratings of kučera and francis words. *Behavior Research Methods*, pages 455—-463.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5:3.

Shah, K., Logacheva, V., Paetzold, G., Blain, F., Beck, D., Bougares, F., and Specia, L. (2015). Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 342–347.

Shalev-Shwartz, S. and Tewari, A. (2009). Stochastic methods for l1 regularized loss minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 929–936, New York, NY, USA. ACM.

Shardlow, M. (2013a). A comparison of techniques to automatically identify complex words. In *Proceedings of the 51st ACL Student Research Workshop*, pages 103–109.

Shardlow, M. (2013b). The cw corpus: A new resource for evaluating the identification of complex words. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77.

Shardlow, M. (2014a). Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *Proceedings of the 9th LREC*, pages 1583–1590.

Shardlow, M. (2014b). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*.

Shelley, M. (2007). *Frankenstein*. Pearson Education.

Siddharthan, A. (2006). Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Siddharthan, A. (2014). A survey of research on text simplification. *International Journal of Applied Linguistics*, pages 259—298.

Sinha, R. (2012). Unt-simprank: Systems for lexical simplification ranking. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, pages 493–496.

Smola, A. and Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, 9:155–161.

sp, s., Kumar, A., and K P, S. (2016). Amritacen at semeval-2016 task 11: Complex word identification using word embedding. In *Proceedings of the 10th SemEval*, pages 1022–1027.

Specia, L. (2010). Translating from complex to simplified sentences. In *Computational Processing of the Portuguese Language*, pages 30–39.

Specia, L., Jauhar, S. K., and Mihalcea, R. (2012). Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 1st SemEval*, pages 347–355.

Specia, L., Paetzold, G., and Scarton, C. (2015). Multi-level translation quality prediction with quest++. In *Proceedings of the 53rd ACL*, pages 115–120.

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. In *Proceedings of the 2002 ICSLP*, pages 257–286.

Stone, V. E., Baron-Cohen, S., Calder, A., Keane, J., and Young, A. (2003). Acquired theory of mind impairments in individuals with bilateral amygdala lesions. *Neuropsychologia*, 41(2):209–220.

Sturgeon, T. (1948). *Not Without Sorcery*. Ballantine books. Ballantine Books.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th ICML*, pages 1017–1024.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. 27:3104–3112.

Swinney, D., Zurif, E., and Nicol, J. (1989). The effects of focal brain damage on sentence processing: An examination of the neurological organization of a mental module. *Journal of Cognitive Neuroscience*, 1(1):25–37.

Thomas, S. R. and Anderson, S. (2012). Wordnet-based lexical simplification of a document. In *Proceedings of 2012 KONVENS*, pages 80–88.

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*, 5:1035–1038.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th ACL*, pages 384–394.

Tweissi, A. I. (1998). The effects of the amount and type of simplification on foreign language reading comprehension. *Reading in a Foreign Language*, 11(2):191–204.

Uhrik, C. and Ward, W. (1997). Confidence metrics based on n-gram language model backoff behaviors. *EUROSPEECH*.

Ulijn, J. M. and Strother, J. B. (1990). The effect of syntactic simplification on reading est texts as l1 and l2. *Journal of research in reading*, 13(1):38–54.

Van Heuven, W. J., Mandera, P., Keuleers, E., and Brysbaert, M. (2014). Subtlex-uk: A new and improved word frequency database for british english. *The Quarterly Journal of Experimental Psychology*, 67:1176–1190.

Vega, F. (2008). *Psicología de la lectura*. Wolters Kluwer Educación.

Vega, F. C., Nosti, M. G., Gutiérrez, A. B., and Brysbaert, M. (2011). Subtlex-esp: Spanish word frequencies based on film subtitles. *Psicológica: Revista de metodología y psicología experimental*, 32:133–143.

Waldron, L., Pintilie, M., Tsao, M.-S., Shepherd, F. A., Huttenhower, C., and Jurisica, I. (2011). Optimized application of penalized regression methods to diverse genomic data. *Bioinformatics*, 27(24):3399–3406.

Watanabe, W. M., Junior, A. C., Uzêda, V. R., Fortes, R. P. d. M., Pardo, T. A. S., and Aluísio, S. M. (2009). Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM*, pages 29–36.

Wauthier, F., Jordan, M., and Jojic, N. (2013). Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning*, pages 109–117.

Wróbel, K. (2016). Plujagh at semeval-2016 task 11: Simple system for complex word identification. In *Proceedings of the 10th SemEval*, pages 953–957.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of ACL*.

Wubben, S., van den Bosch, A., and Krahmer, E. (2012). Sentence simplification by monolingual machine translation. In *Proceedings of the 50th ACL*, pages 1015–1024.

Yaneva, V., Temnikova, I., and Mitkov, R. (2015). Accessible texts for autism: An eye-tracking study. In *Proceedings of the 17th SIGACCESS*, pages 49–57.

Yano, Y., Long, M. H., and Ross, S. (1994). The effects of simplified and elaborated texts on foreign language reading comprehension. *Language learning*, 44(2):189–219.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd ACL*, pages 189–196. Association for Computational Linguistics.

Yatskar, M., Pang, B., Danescu-Niculescu-Mizil, C., and Lee, L. (2010). For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Yih, W.-t. and Qazvinian, V. (2012). Measuring word relatedness using heterogeneous vector space models. In *Proceedings of the 2012 NAACL*, pages 616–620.

Young, D. N. (1999). Linguistic simplification of sl reading material: Effective instructional practice? *The Modern Language Journal*, 83(3):350–366.

Zampieri, M., Tan, L., and van Genabith, J. (2016). Macsaar at semeval-2016 task 11: Zipfian and character features for complexword identification. In *Proceedings of the 10th SemEval*, pages 1001–1005.

Zevin, J. D. and Seidenberg, M. S. (2002). Age of acquisition effects in word reading and other tasks. *Journal of Memory and language*, 47(1):1–29.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67:301–320.

Zou, W. Y., Socher, R., Cer, D. M., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 EMNLP*, pages 1393–1398.

Appendix A

Binary Classifiers in Lexical Simplification

In this experiment, we evaluate how well the most successful binary classification approaches in the experiments of Section 3.2 perform as part of an LS system.

A.1 Approaches

We include three binary classifiers in our evaluation:

- **Grammaticality Identifier (Grammaticality)**: Selects only those candidates that are classified as grammatical by a Decision Tree binary classifier.
- **Meaning Preservation Identifier (Meaning Preservation)**: Selects only those candidates that are classified as meaning preserving by a Decision Tree binary classifier.
- **Appropriateness Identifier (Appropriateness)**: Selects only those candidates that are classified to be both grammatical and meaning preserving by a Decision Tree binary classifier.

All three identifiers were trained over the concatenation of the *optimistic* training set and the test set of their respective word replacement properties. The hyper-parameters of each identifier are maximised through 10-fold cross-validation. We compare their performance with that of six baselines:

- **First Sense (First)**: Selects words which are listed as synonyms under the first WordNet sense of the target word.

- **Random Sense (Random)**: Selects words which are listed as synonyms under a random WordNet sense of the target word.
- **Lesk Algorithm (Lesk)**: Uses the algorithm introduced by Lesk (1986), described in Section 5.4.
- **Path Similarity (Path)**: Uses the algorithm introduced by Leacock and Chodorow (1998), described in Section 5.4.
- **Word Clustering (Belder)**: Uses a strategy similar to the one presented by De Belder and Moens (2010), described in Section 5.4.
- **Co-Occurrence Model Filtering (Biran)**: Uses the strategy introduced by Biran et al. (2011), described in Section 5.4.

In order to evaluate their performance in practice, we employ the aforementioned strategies in selecting candidate substitutions produced by various Substitution Generation systems. The generators included in this experiment are:

- **Devlin** (Devlin and Tait, 1998): Generates candidate substitutions by extracting synonyms from WordNet, as described in Section 5.4.
- **Kauchak** (Horn et al., 2014): Generates candidate substitutions from complex-to-simple parallel corpora, as described in Section 5.4.
- **Yamamoto** (Kajiwara et al., 2013): Produces candidate substitutions by querying dictionaries for target words, retrieving example and/or definition sentences, and then extracting any words that share the same POS tag as the target word. For this approach, we use the Merriam Dictionary and Thesaurus API[1]. To tag the dictionary examples and definitions, we use the Stanford Parser (Klein and Manning, 2003).
- **All**: Combines the substitutions generated by all other generators.

In our full pipeline evaluation, we pair our selectors with various metric-based Substitution Ranking approaches. They metrics used by them are:

- **Frequency**: Candidates are ranked according to their frequency in the Simple Wikipedia corpus (Kauchak, 2013). The more frequent the word, the simpler it is.

[1] http://www.merriam-webster.com

- **Length**: Candidates are ranked according to their length. The less characters a candidate have, the simpler it is.
- **Senses**: Candidates are ranked according to their number of senses in WordNet. The more senses a word has, the simpler it is.
- **Synonyms**: Candidates are ranked according to their number of synonyms in WordNet. The more synonyms a word has, the simpler it is.
- **Hypernyms**: Candidates are ranked according to their number of hypernyms in WordNet. The more hypernyms a word has, the simpler it is.
- **Hyponyms**: Candidates are ranked according to their number of hyponyms in WordNet. The more hyponyms a word has, the simpler it is.

A.2 Datasets

As a gold-standard for our selectors, we use the NNSeval dataset, described in Section 8.4.

A.3 Metrics

To assess the performance of our identifiers in Substitution Selection alone, we use the metrics proposed by Paetzold (2015b), which are:

- **Potential**: The proportion of instances in which at least one of the substitutions selected is present in the gold-standard.
- **Precision**: The proportion of selected substitutions that are present in the gold-standard.
- **Recall**: The proportion of gold-standard substitutions that are included in the selected substitutions.
- **F1**: The harmonic mean between Precision and Recall.

For our full pipeline evaluation, in which our identifiers are paired with substitution generators and rankers to compose full LS systems, we use the metrics proposed by Horn et al. (2014), which are:

- **Precision**: The proportion of instances in which the target word was replaced with any of the candidates in the dataset, including the target word itself.

- **Accuracy**: The proportion of instances in which the target word was replaced with any of the candidates in the dataset, except for the target word itself.

- **Changed Proportion**: The proportion of times in which the target word was replaced with a different word.

A.4 Substitution Selection Evaluation

The performance obtained by all identifiers and selectors with respect to each generator is illustrated in Tables A.1 through A.4. The Grammaticality Identifier is the one to perform best across all scenarios evaluated. Nonetheless, our identifiers are not very effective. Even though they were able to obtain higher F1 scores than other selectors in most scenarios, they are often less effective than not performing selection at all. The fact that the highest F1 score overall is lower than 0.1 further highlight the need for better SG and SS approaches for non-native speakers of English.

Selector	Potential	Precision	Recall	F1
First	0.033	0.020	0.005	0.008
Lesk	0.109	0.030	0.018	0.022
Path	0.013	0.012	0.002	0.004
Biran	0.201	0.052	0.044	0.047
Belder	0.126	**0.167**	0.018	0.033
Grammaticality	0.368	0.062	0.075	**0.068**
Meaning Preservation	0.322	0.064	0.055	0.059
Appropriateness	0.251	0.070	0.039	0.050
No Selection	**0.473**	0.046	**0.106**	0.065

Table A.1 Evaluation results for SS approaches with respect to substitutions generated by the Merriam generator

Selector	Potential	Precision	Recall	F1
First	0.004	0.009	0.001	0.001
Lesk	0.021	0.021	0.003	0.006
Path	0.000	0.000	0.000	0.000
Biran	0.146	0.028	0.027	0.027
Belder	0.038	**0.205**	0.005	0.010
Grammaticality	0.255	0.038	0.040	**0.039**
Meaning Preservation	0.184	0.034	0.028	0.031
Appropriateness	0.126	0.036	0.018	0.024
No Selection	**0.314**	0.026	**0.061**	0.037

Table A.2 Evaluation results for SS approaches with respect to substitutions generated by the Yamamoto generator

Selector	Potential	Precision	Recall	F1
First	0.059	0.078	0.009	0.017
Lesk	0.109	0.061	0.016	0.026
Path	0.008	0.042	0.002	0.003
Biran	0.226	0.113	0.042	0.062
Belder	0.130	**0.318**	0.019	0.036
Grammaticality	0.356	0.111	0.065	0.082
Meaning Preservation	0.268	0.119	0.046	0.066
Appropriateness	0.209	0.125	0.036	0.056
No Selection	**0.485**	0.092	**0.093**	**0.092**

Table A.3 Evaluation results for SS approaches with respect to substitutions generated by the Devlin generator

Selector	Potential	Precision	Recall	F1
First	0.042	0.078	0.006	0.010
Lesk	0.084	0.090	0.012	0.022
Path	0.004	0.018	0.001	0.001
Biran	0.226	0.109	0.039	0.057
Belder	0.063	**0.283**	0.008	0.016
Grammaticality	0.339	0.120	0.059	0.079
Meaning Preservation	0.247	0.115	0.040	0.060
Appropriateness	0.192	0.124	0.030	0.049
No Selection	**0.414**	0.084	**0.079**	**0.081**

Table A.4 Evaluation results for SS approaches with respect to substitutions generated by the Biran generator

Selector	Potential	Precision	Recall	F1
First	0.025	0.019	0.003	0.006
Lesk	0.050	0.033	0.008	0.013
Path	0.004	0.004	0.001	0.001
Biran	0.251	0.103	0.049	0.067
Belder	0.130	**0.276**	0.019	0.036
Grammaticality	0.381	0.094	0.070	0.080
Meaning Preservation	0.276	0.094	0.044	0.060
Appropriateness	0.238	0.114	0.038	0.057
No Selection	**0.464**	0.134	**0.088**	0.106

Table A.5 Evaluation results for SS approaches with respect to substitutions generated by the Kauchak generator

Selector	Potential	Precision	Recall	F1
First	0.042	0.027	0.006	0.010
Lesk	0.151	0.049	0.023	0.032
Path	0.013	0.008	0.002	0.003
Biran	0.289	0.114	0.059	0.078
Belder	0.234	**0.225**	0.035	0.060
Grammaticality	0.498	0.112	0.094	0.103
Meaning Preservation	0.364	0.116	0.062	0.081
Appropriateness	0.339	0.129	0.055	0.077
No Selection	**0.661**	0.105	**0.141**	0.121

Table A.6 Evaluation results for SS approaches with respect to substitutions generated by the Glavas generator

Selector	Potential	Precision	Recall	F1
First	0.054	0.043	0.008	0.013
Lesk	0.176	0.060	0.026	0.037
Path	0.013	0.011	0.002	0.003
Biran	0.322	0.122	0.068	0.087
Belder	0.247	**0.201**	0.034	0.058
Grammaticality	0.590	0.125	0.112	0.118
Meaning Preservation	0.431	0.139	0.078	0.100
Appropriateness	0.402	0.162	0.070	0.098
No Selection	**0.699**	0.118	**0.161**	0.136

Table A.7 Evaluation results for SS approaches with respect to substitutions generated by the Paetzold generator

Selector	Potential	Precision	Recall	F1
First	0.075	0.015	0.013	0.014
Lesk	0.285	0.028	0.050	0.036
Path	0.021	0.005	0.003	0.004
Biran	0.502	0.051	0.161	0.077
Belder	0.339	**0.183**	0.054	0.083
Grammaticality	0.854	0.054	0.238	0.089
Meaning Preservation	0.736	0.054	0.168	0.082
Appropriateness	0.653	0.063	0.137	0.086
No Selection	**0.962**	0.043	**0.356**	0.076

Table A.8 Evaluation results for SS approaches with respect to substitutions generated by all generators combined

A.5 Full Pipeline Evaluation

For completion, we also evaluated how our identifiers perform when part of a full LS system. Tables A.9 through A.17 report on the results obtained when pairing all selectors assessed in our previous experiment with the metric-based rankers described in Section A.1. Given their high Potential and competitive F1, we use the candidates produced by all previously mentioned generators combined.

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.017	0.013	0.996
Word Length	0.050	0.050	1.000
Senses	0.059	0.059	1.000
Synonyms	0.038	0.038	1.000
Hypernyms	0.050	0.050	1.000
Hyponyms	0.038	0.038	1.000

Table A.9 Full pipeline scores with respect to substitutions generated by all generators combined, without any selection

Although in practice our identifiers have shown to be a better alternative than not performing selection at all, the Belder selector, which employs an unsupervised strategy, have outperformed all of them in Precision and Accuracy.

Given the results, we can conclude that our Grammaticality, Meaning Preservation and Appropriateness identifiers are not reliable Substitution Selection approaches. Considering the performance obtained by the Belder selector, the cost inherent to producing annotated data, and the fact that annotators tend to have different perceptions and standards for linguistic

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.004	0.004	1.000
Word Length	0.021	0.017	0.996
Senses	0.201	0.042	0.841
Synonyms	0.197	0.046	0.849
Hypernyms	0.163	0.050	0.887
Hyponyms	0.172	0.054	0.883

Table A.10 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.008	0.008	1.000
Word Length	0.050	0.050	1.000
Senses	0.159	0.067	0.908
Synonyms	0.138	0.054	0.916
Hypernyms	0.109	0.038	0.929
Hyponyms	0.096	0.033	0.937

Table A.11 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.004	0.004	1.000
Word Length	0.029	0.004	0.975
Senses	0.230	0.013	0.782
Synonyms	0.230	0.013	0.782
Hypernyms	0.226	0.013	0.787
Hyponyms	0.213	0.013	0.799

Table A.12 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.013	0.013	1.000
Word Length	0.046	0.046	1.000
Senses	0.050	0.050	1.000
Synonyms	0.054	0.050	0.996
Hypernyms	0.042	0.042	1.000
Hyponyms	0.025	0.021	0.996

Table A.13 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.289	0.142	0.854
Word Length	0.314	0.197	0.883
Senses	0.326	0.218	0.891
Synonyms	0.314	0.184	0.870
Hypernyms	0.297	0.192	0.895
Hyponyms	0.339	0.201	0.862

Table A.14 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.013	0.013	1.000
Word Length	0.059	0.059	1.000
Senses	0.096	0.096	1.000
Synonyms	0.109	0.109	1.000
Hypernyms	0.079	0.079	1.000
Hyponyms	0.079	0.079	1.000

Table A.15 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Grammaticality Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.038	0.038	1.000
Word Length	0.046	0.046	1.000
Senses	0.079	0.079	1.000
Synonyms	0.071	0.071	1.000
Hypernyms	0.075	0.075	1.000
Hyponyms	0.084	0.084	1.000

Table A.16 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Meaning Preservation Selector

Ranker	Precision	Accuracy	Changed Proportion
Frequency	0.042	0.042	1.000
Word Length	0.054	0.054	1.000
Senses	0.092	0.088	0.996
Synonyms	0.096	0.084	0.987
Hypernyms	0.100	0.092	0.992
Hyponyms	0.092	0.088	0.996

Table A.17 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Appropriateness Selector

conventions, we hypothesise that further exploring unsupervised approaches for SS may be more promising than investing in supervised binary classifiers.

Appendix B

Performance Comparison of Word Embedding Models in Substitution Generation

We present all results obtained in the performance comparison presented in Section 5.5. The scores are illustrated in Tables B.1 through B.16.

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.610	0.654	0.668	0.548	0.570	0.590
10	0.730	0.768	0.784	0.634	0.690	0.660
15	0.788	0.812	0.818	0.702	0.710	0.696
20	0.812	0.836	0.842	0.714	0.728	0.716
25	0.824	0.854	**0.862**	0.732	0.738	0.726

Table B.1 Potential results for traditional embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.672	0.710	0.716	0.630	0.660	0.668
10	0.796	0.830	0.834	0.722	0.756	0.758
15	0.834	0.858	0.872	0.772	0.806	0.806
20	0.862	0.884	0.882	0.792	0.820	0.824
25	0.868	0.906	**0.908**	0.812	0.834	0.826

Table B.2 Potential results for retrofitted embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.672	0.708	0.728	0.596	0.626	0.628
10	0.778	0.802	0.828	0.682	0.690	0.686
15	0.824	0.850	0.862	0.708	0.724	0.714
20	0.844	0.866	0.876	0.726	0.746	0.740
25	0.858	0.880	**0.892**	0.732	0.754	0.742

Table B.3 Potential results for context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.736	0.754	0.766	0.702	0.726	0.720
10	0.836	0.852	0.862	0.778	0.798	0.796
15	0.870	0.884	0.890	0.804	0.826	0.822
20	0.890	0.902	0.912	0.824	0.836	0.830
25	0.896	0.906	**0.920**	0.828	0.846	0.836

Table B.4 Potential results for retrofitted context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.194	0.212	**0.218**	0.172	0.182	0.185
10	0.136	0.155	0.157	0.117	0.129	0.129
15	0.112	0.121	0.124	0.093	0.098	0.099
20	0.092	0.101	0.104	0.077	0.081	0.080
25	0.079	0.087	0.090	0.065	0.068	0.069

Table B.5 Precision results for traditional embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.210	0.225	**0.230**	0.200	0.213	0.221
10	0.154	0.168	0.170	0.144	0.154	0.157
15	0.122	0.132	0.135	0.114	0.122	0.122
20	0.101	0.111	0.114	0.095	0.100	0.102
25	0.089	0.098	0.100	0.082	0.086	0.086

Table B.6 Precision results for retrofitted embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.222	0.245	**0.256**	0.201	0.222	0.227
10	0.159	0.177	0.185	0.149	0.161	0.164
15	0.131	0.142	0.147	0.123	0.134	0.135
20	0.110	0.118	0.124	0.107	0.116	0.119
25	0.096	0.105	0.109	0.098	0.106	0.107

Table B.7 Precision results for context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.253	0.263	**0.269**	0.238	0.249	0.249
10	0.179	0.192	0.197	0.172	0.183	0.183
15	0.143	0.153	0.157	0.141	0.149	0.149
20	0.122	0.129	0.133	0.120	0.127	0.128
25	0.106	0.113	0.117	0.108	0.114	0.115

Table B.8 Precision results for retrofitted context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.078	0.085	0.088	0.070	0.073	0.075
10	0.110	0.125	0.126	0.094	0.104	0.104
15	0.135	0.146	0.150	0.112	0.118	0.120
20	0.149	0.163	0.168	0.125	0.131	0.130
25	0.159	0.176	**0.181**	0.132	0.138	0.138

Table B.9 Recall results for traditional embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.085	0.091	0.093	0.081	0.086	0.089
10	0.124	0.135	0.137	0.116	0.124	0.127
15	0.148	0.160	0.164	0.138	0.148	0.148
20	0.164	0.178	0.183	0.152	0.161	0.164
25	0.179	0.197	**0.201**	0.165	0.173	0.173

Table B.10 Recall results for retrofitted embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.089	0.098	0.102	0.079	0.087	0.089
10	0.127	0.140	0.147	0.113	0.121	0.122
15	0.156	0.168	0.174	0.133	0.142	0.141
20	0.172	0.184	0.193	0.145	0.153	0.154
25	0.185	0.201	**0.209**	0.153	0.161	0.159

Table B.11 Recall results for context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.101	0.105	0.107	0.094	0.098	0.098
10	0.143	0.153	0.157	0.134	0.141	0.140
15	0.171	0.182	0.187	0.159	0.166	0.166
20	0.192	0.205	0.210	0.175	0.182	0.182
25	0.208	0.221	**0.229**	0.188	0.196	0.195

Table B.12 Recall results for retrofitted context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.111	0.122	0.125	0.099	0.104	0.106
10	0.122	0.138	**0.140**	0.104	0.115	0.115
15	0.122	0.132	0.136	0.101	0.107	0.109
20	0.114	0.124	0.129	0.095	0.100	0.099
25	0.105	0.117	0.120	0.087	0.091	0.092

Table B.13 F1 results for traditional embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.121	0.129	0.132	0.115	0.123	0.127
10	0.137	0.150	**0.152**	0.128	0.138	0.141
15	0.134	0.145	0.148	0.125	0.134	0.134
20	0.125	0.137	0.140	0.117	0.123	0.125
25	0.119	0.131	0.133	0.109	0.114	0.115

Table B.14 F1 results for retrofitted embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.127	0.140	0.146	0.113	0.125	0.128
10	0.141	0.156	**0.164**	0.129	0.138	0.140
15	0.142	0.153	0.159	0.128	0.138	0.138
20	0.135	0.144	0.151	0.123	0.132	0.134
25	0.127	0.138	0.143	0.120	0.128	0.128

Table B.15 F1 results for context-aware embedding models

	CBOW			Skip-Gram		
#	300	500	700	300	500	700
5	0.144	0.150	0.154	0.135	0.141	0.141
10	0.159	0.170	**0.175**	0.151	0.160	0.159
15	0.156	0.166	0.171	0.149	0.157	0.157
20	0.149	0.159	0.163	0.143	0.149	0.150
25	0.141	0.150	0.154	0.137	0.144	0.145

Table B.16 F1 results for retrofitted context-aware embedding models

Appendix C

Benchmarking Results

The complete results of the full pipeline benchmark described in Section 8.5.1 are illustrated in Tables C.1 through C.56.

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.046	0.042	0.996
Syllable Count	0.067	0.063	0.996
Senses	0.059	0.059	1.000
Synonyms	0.042	0.038	0.996
Hypernyms	0.050	0.050	1.000
Hyponyms	0.038	0.038	1.000
Frequency: Simple Wiki	0.017	0.013	0.996
Frequency: Brown	0.079	0.079	1.000
Frequency: SUBTLEX	0.008	0.004	0.996
Frequency: SubIMDB	0.054	0.054	1.000
N-gram: Simple Wiki	0.339	0.222	0.883
N-gram: Brown	0.209	0.130	0.921
N-gram: SUBTLEX	0.226	0.151	0.925
N-gram: SubIMDB	0.285	0.197	0.912
Age of Acquisition	0.021	0.021	1.000
Familiarity	0.054	0.054	1.000
Concreteness	0.038	0.038	1.000
Imagery	0.029	0.029	1.000
Biran Ranker	0.046	0.042	0.996
Bott Ranker	0.067	0.067	1.000
Yamamoto Ranker	0.109	0.059	0.950
Horn Ranker	0.180	0.172	0.992
Glavas Ranker	0.310	0.121	0.812
Boundary Ranker	0.130	0.100	0.971

Table C.1 Full pipeline scores with respect to substitutions generated by all generators combined, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.226	0.013	0.787
Syllable Count	0.259	0.013	0.753
Senses	0.753	0.029	0.276
Synonyms	0.753	0.038	0.285
Hypernyms	0.552	0.042	0.490
Hyponyms	0.485	0.042	0.556
Frequency: Simple Wiki	0.021	0.004	0.983
Frequency: Brown	0.510	0.025	0.515
Frequency: SUBTLEX	0.025	0.004	0.979
Frequency: SubIMDB	0.314	0.033	0.720
N-gram: Simple Wiki	0.816	0.033	0.218
N-gram: Brown	0.561	0.025	0.464
N-gram: SUBTLEX	0.674	0.033	0.360
N-gram: SubIMDB	0.582	0.025	0.444
Age of Acquisition	0.025	0.008	0.983
Familiarity	0.741	0.029	0.289
Concreteness	0.720	0.046	0.326
Imagery	0.787	0.046	0.259
Biran Ranker	0.226	0.008	0.782
Bott Ranker	0.561	0.029	0.469
Yamamoto Ranker	0.686	0.033	0.347
Horn Ranker	0.736	0.017	0.280
Glavas Ranker	0.724	0.025	0.301
Boundary Ranker	0.435	0.033	0.598

Table C.2 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.201	0.046	0.845
Syllable Count	0.226	0.038	0.812
Senses	0.490	0.050	0.561
Synonyms	0.444	0.046	0.603
Hypernyms	0.377	0.038	0.661
Hyponyms	0.331	0.033	0.703
Frequency: Simple Wiki	0.025	0.008	0.983
Frequency: Brown	0.368	0.063	0.695
Frequency: SUBTLEX	0.013	0.004	0.992
Frequency: SubIMDB	0.201	0.054	0.854
N-gram: Simple Wiki	0.632	0.071	0.439
N-gram: Brown	0.485	0.054	0.569
N-gram: SUBTLEX	0.456	0.067	0.611
N-gram: SubIMDB	0.473	0.046	0.573
Age of Acquisition	0.021	0.013	0.992
Familiarity	0.368	0.067	0.699
Concreteness	0.280	0.046	0.766
Imagery	0.318	0.046	0.728
Biran Ranker	0.167	0.033	0.866
Bott Ranker	0.381	0.059	0.678
Yamamoto Ranker	0.506	0.042	0.536
Horn Ranker	0.594	0.054	0.460
Glavas Ranker	0.632	0.063	0.431
Boundary Ranker	0.331	0.054	0.724

Table C.3 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.310	0.000	0.690
Syllable Count	0.335	0.000	0.665
Senses	0.904	0.000	0.096
Synonyms	0.904	0.000	0.096
Hypernyms	0.678	0.000	0.322
Hyponyms	0.598	0.000	0.402
Frequency: Simple Wiki	0.029	0.004	0.975
Frequency: Brown	0.536	0.004	0.469
Frequency: SUBTLEX	0.029	0.004	0.975
Frequency: SubIMDB	0.356	0.004	0.649
N-gram: Simple Wiki	0.849	0.004	0.155
N-gram: Brown	0.619	0.004	0.385
N-gram: SUBTLEX	0.715	0.013	0.297
N-gram: SubIMDB	0.619	0.000	0.381
Age of Acquisition	0.029	0.000	0.971
Familiarity	0.837	0.004	0.167
Concreteness	0.824	0.004	0.180
Imagery	0.879	0.004	0.126
Biran Ranker	0.259	0.000	0.741
Bott Ranker	0.649	0.008	0.360
Yamamoto Ranker	0.736	0.004	0.268
Horn Ranker	0.787	0.008	0.222
Glavas Ranker	0.720	0.000	0.280
Boundary Ranker	0.473	0.004	0.531

Table C.4 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.469	0.042	0.573
Syllable Count	0.485	0.050	0.565
Senses	0.481	0.050	0.569
Synonyms	0.481	0.050	0.569
Hypernyms	0.464	0.042	0.577
Hyponyms	0.444	0.021	0.577
Frequency: Simple Wiki	0.427	0.017	0.590
Frequency: Brown	0.485	0.063	0.577
Frequency: SUBTLEX	0.427	0.013	0.586
Frequency: SubIMDB	0.473	0.063	0.590
N-gram: Simple Wiki	0.649	0.146	0.498
N-gram: Brown	0.565	0.088	0.523
N-gram: SUBTLEX	0.586	0.096	0.510
N-gram: SubIMDB	0.615	0.126	0.510
Age of Acquisition	0.431	0.025	0.594
Familiarity	0.490	0.059	0.569
Concreteness	0.448	0.021	0.573
Imagery	0.452	0.025	0.573
Biran Ranker	0.460	0.038	0.577
Bott Ranker	0.473	0.050	0.577
Yamamoto Ranker	0.527	0.050	0.523
Horn Ranker	0.565	0.096	0.531
Glavas Ranker	0.628	0.088	0.460
Boundary Ranker	0.523	0.088	0.565

Table C.5 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.695	0.126	0.431
Syllable Count	0.724	0.142	0.418
Senses	0.699	0.134	0.435
Synonyms	0.674	0.138	0.464
Hypernyms	0.603	0.134	0.531
Hyponyms	0.682	0.126	0.444
Frequency: Simple Wiki	0.498	0.084	0.586
Frequency: Brown	0.674	0.121	0.448
Frequency: SUBTLEX	0.611	0.130	0.519
Frequency: SubIMDB	0.649	0.130	0.481
N-gram: Simple Wiki	0.749	0.142	0.393
N-gram: Brown	0.728	0.121	0.393
N-gram: SUBTLEX	0.703	0.130	0.427
N-gram: SubIMDB	0.736	0.121	0.385
Age of Acquisition	0.644	0.117	0.473
Familiarity	0.661	0.126	0.464
Concreteness	0.674	0.096	0.423
Imagery	0.636	0.117	0.481
Biran Ranker	0.703	0.155	0.452
Bott Ranker	0.715	0.134	0.418
Yamamoto Ranker	0.820	0.109	0.289
Horn Ranker	0.820	0.134	0.314
Glavas Ranker	0.870	0.105	0.234
Boundary Ranker	0.707	0.138	0.431

Table C.6 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.109	0.105	0.996
Syllable Count	0.163	0.138	0.975
Senses	0.105	0.096	0.992
Synonyms	0.105	0.096	0.992
Hypernyms	0.105	0.092	0.987
Hyponyms	0.100	0.079	0.979
Frequency: Simple Wiki	0.075	0.046	0.971
Frequency: Brown	0.088	0.088	1.000
Frequency: SUBTLEX	0.113	0.054	0.941
Frequency: SubIMDB	0.096	0.096	1.000
N-gram: Simple Wiki	0.339	0.222	0.883
N-gram: Brown	0.251	0.167	0.916
N-gram: SUBTLEX	0.243	0.155	0.912
N-gram: SubIMDB	0.293	0.201	0.908
Age of Acquisition	0.038	0.038	1.000
Familiarity	0.079	0.079	1.000
Concreteness	0.130	0.096	0.967
Imagery	0.138	0.130	0.992
Biran Ranker	0.113	0.092	0.979
Bott Ranker	0.113	0.096	0.983
Yamamoto Ranker	0.234	0.079	0.845
Horn Ranker	0.201	0.142	0.941
Glavas Ranker	0.351	0.134	0.782
Boundary Ranker	0.163	0.134	0.971

Table C.7 Full pipeline scores with respect to substitutions generated by all generators combined, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.226	0.226	1.000
Syllable Count	0.134	0.134	1.000
Senses	0.138	0.138	1.000
Synonyms	0.159	0.159	1.000
Hypernyms	0.163	0.163	1.000
Hyponyms	0.146	0.146	1.000
Frequency: Simple Wiki	0.029	0.029	1.000
Frequency: Brown	0.259	0.259	1.000
Frequency: SUBTLEX	0.021	0.021	1.000
Frequency: SubIMDB	0.305	0.305	1.000
N-gram: Simple Wiki	0.272	0.272	1.000
N-gram: Brown	0.276	0.276	1.000
N-gram: SUBTLEX	0.285	0.285	1.000
N-gram: SubIMDB	0.272	0.272	1.000
Age of Acquisition	0.079	0.079	1.000
Familiarity	0.309	0.309	1.000
Concreteness	0.146	0.146	1.000
Imagery	0.222	0.222	1.000
Biran Ranker	0.230	0.230	1.000
Bott Ranker	0.280	0.280	1.000
Yamamoto Ranker	0.255	0.255	1.000
Horn Ranker	0.498	0.230	0.732
Glavas Ranker	0.506	0.251	0.745
Boundary Ranker	0.297	0.297	1.000

Table C.8 Full pipeline scores with respect to substitutions generated by the Paetzold generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.046	0.046	1.000
Syllable Count	0.042	0.042	1.000
Senses	0.046	0.046	1.000
Synonyms	0.046	0.046	1.000
Hypernyms	0.042	0.042	1.000
Hyponyms	0.042	0.042	1.000
Frequency: Simple Wiki	0.025	0.025	1.000
Frequency: Brown	0.042	0.042	1.000
Frequency: SUBTLEX	0.025	0.025	1.000
Frequency: SubIMDB	0.042	0.042	1.000
N-gram: Simple Wiki	0.046	0.046	1.000
N-gram: Brown	0.046	0.046	1.000
N-gram: SUBTLEX	0.046	0.046	1.000
N-gram: SubIMDB	0.042	0.042	1.000
Age of Acquisition	0.038	0.038	1.000
Familiarity	0.046	0.046	1.000
Concreteness	0.046	0.046	1.000
Imagery	0.046	0.046	1.000
Biran Ranker	0.042	0.042	1.000
Bott Ranker	0.046	0.046	1.000
Yamamoto Ranker	0.046	0.046	1.000
Horn Ranker	0.971	0.008	0.038
Glavas Ranker	0.933	0.021	0.088
Boundary Ranker	0.042	0.042	1.000

Table C.9 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.067	0.067	1.000
Syllable Count	0.063	0.063	1.000
Senses	0.059	0.059	1.000
Synonyms	0.059	0.059	1.000
Hypernyms	0.050	0.050	1.000
Hyponyms	0.054	0.054	1.000
Frequency: Simple Wiki	0.033	0.033	1.000
Frequency: Brown	0.100	0.100	1.000
Frequency: SUBTLEX	0.029	0.029	1.000
Frequency: SubIMDB	0.088	0.088	1.000
N-gram: Simple Wiki	0.100	0.100	1.000
N-gram: Brown	0.084	0.084	1.000
N-gram: SUBTLEX	0.105	0.105	1.000
N-gram: SubIMDB	0.088	0.088	1.000
Age of Acquisition	0.059	0.059	1.000
Familiarity	0.084	0.084	1.000
Concreteness	0.075	0.075	1.000
Imagery	0.075	0.075	1.000
Biran Ranker	0.067	0.067	1.000
Bott Ranker	0.092	0.092	1.000
Yamamoto Ranker	0.084	0.084	1.000
Horn Ranker	0.849	0.038	0.188
Glavas Ranker	0.707	0.059	0.351
Boundary Ranker	0.100	0.100	1.000

Table C.10 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.013	0.013	1.000
Syllable Count	0.013	0.013	1.000
Senses	0.013	0.013	1.000
Synonyms	0.013	0.013	1.000
Hypernyms	0.013	0.013	1.000
Hyponyms	0.013	0.013	1.000
Frequency: Simple Wiki	0.008	0.008	1.000
Frequency: Brown	0.013	0.013	1.000
Frequency: SUBTLEX	0.008	0.008	1.000
Frequency: SubIMDB	0.013	0.013	1.000
N-gram: Simple Wiki	0.008	0.008	1.000
N-gram: Brown	0.013	0.013	1.000
N-gram: SUBTLEX	0.008	0.008	1.000
N-gram: SubIMDB	0.013	0.013	1.000
Age of Acquisition	0.004	0.004	1.000
Familiarity	0.013	0.013	1.000
Concreteness	0.013	0.013	1.000
Imagery	0.013	0.013	1.000
Biran Ranker	0.008	0.008	1.000
Bott Ranker	0.008	0.008	1.000
Yamamoto Ranker	0.008	0.008	1.000
Horn Ranker	0.987	0.000	0.013
Glavas Ranker	0.950	0.000	0.050
Boundary Ranker	0.013	0.013	1.000

Table C.11 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.100	0.100	1.000
Syllable Count	0.084	0.084	1.000
Senses	0.084	0.084	1.000
Synonyms	0.088	0.088	1.000
Hypernyms	0.088	0.088	1.000
Hyponyms	0.075	0.075	1.000
Frequency: Simple Wiki	0.029	0.029	1.000
Frequency: Brown	0.113	0.113	1.000
Frequency: SUBTLEX	0.025	0.025	1.000
Frequency: SubIMDB	0.167	0.167	1.000
N-gram: Simple Wiki	0.142	0.142	1.000
N-gram: Brown	0.130	0.130	1.000
N-gram: SUBTLEX	0.155	0.155	1.000
N-gram: SubIMDB	0.142	0.142	1.000
Age of Acquisition	0.075	0.075	1.000
Familiarity	0.155	0.155	1.000
Concreteness	0.088	0.088	1.000
Imagery	0.113	0.113	1.000
Biran Ranker	0.100	0.100	1.000
Bott Ranker	0.126	0.126	1.000
Yamamoto Ranker	0.134	0.134	1.000
Horn Ranker	0.732	0.105	0.372
Glavas Ranker	0.741	0.113	0.372
Boundary Ranker	0.163	0.163	1.000

Table C.12 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.126	0.126	1.000
Syllable Count	0.130	0.130	1.000
Senses	0.155	0.155	1.000
Synonyms	0.151	0.151	1.000
Hypernyms	0.151	0.151	1.000
Hyponyms	0.151	0.151	1.000
Frequency: Simple Wiki	0.134	0.134	1.000
Frequency: Brown	0.134	0.134	1.000
Frequency: SUBTLEX	0.130	0.130	1.000
Frequency: SubIMDB	0.138	0.138	1.000
N-gram: Simple Wiki	0.142	0.142	1.000
N-gram: Brown	0.146	0.146	1.000
N-gram: SUBTLEX	0.151	0.151	1.000
N-gram: SubIMDB	0.134	0.134	1.000
Age of Acquisition	0.126	0.126	1.000
Familiarity	0.138	0.138	1.000
Concreteness	0.121	0.121	1.000
Imagery	0.172	0.172	1.000
Biran Ranker	0.126	0.126	1.000
Bott Ranker	0.130	0.130	1.000
Yamamoto Ranker	0.134	0.134	1.000
Horn Ranker	0.874	0.084	0.209
Glavas Ranker	0.799	0.096	0.297
Boundary Ranker	0.138	0.138	1.000

Table C.13 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.272	0.272	1.000
Syllable Count	0.218	0.218	1.000
Senses	0.255	0.255	1.000
Synonyms	0.218	0.218	1.000
Hypernyms	0.226	0.226	1.000
Hyponyms	0.205	0.205	1.000
Frequency: Simple Wiki	0.109	0.109	1.000
Frequency: Brown	0.280	0.280	1.000
Frequency: SUBTLEX	0.079	0.079	1.000
Frequency: SubIMDB	0.310	0.310	1.000
N-gram: Simple Wiki	0.285	0.285	1.000
N-gram: Brown	0.276	0.276	1.000
N-gram: SUBTLEX	0.285	0.285	1.000
N-gram: SubIMDB	0.272	0.272	1.000
Age of Acquisition	0.109	0.109	1.000
Familiarity	0.309	0.309	1.000
Concreteness	0.159	0.159	1.000
Imagery	0.247	0.247	1.000
Biran Ranker	0.272	0.272	1.000
Bott Ranker	0.276	0.276	1.000
Yamamoto Ranker	0.280	0.280	1.000
Horn Ranker	0.490	0.218	0.728
Glavas Ranker	0.485	0.238	0.753
Boundary Ranker	0.297	0.297	1.000

Table C.14 Full pipeline scores with respect to substitutions generated by the Paetzold generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.259	0.088	0.828
Syllable Count	0.293	0.109	0.816
Senses	0.289	0.071	0.782
Synonyms	0.209	0.075	0.866
Hypernyms	0.192	0.088	0.895
Hyponyms	0.213	0.079	0.866
Frequency: Simple Wiki	0.176	0.021	0.845
Frequency: Brown	0.272	0.075	0.803
Frequency: SUBTLEX	0.192	0.013	0.820
Frequency: SubIMDB	0.226	0.063	0.837
N-gram: Simple Wiki	0.573	0.100	0.527
N-gram: Brown	0.452	0.071	0.619
N-gram: SUBTLEX	0.460	0.105	0.644
N-gram: SubIMDB	0.469	0.109	0.640
Age of Acquisition	0.063	0.063	1.000
Familiarity	0.084	0.084	1.000
Concreteness	0.063	0.054	0.992
Imagery	0.067	0.067	1.000
Biran Ranker	0.226	0.075	0.849
Bott Ranker	0.289	0.067	0.778
Yamamoto Ranker	0.418	0.067	0.649
Horn Ranker	0.544	0.096	0.552
Glavas Ranker	0.552	0.088	0.536
Boundary Ranker	0.318	0.067	0.749

Table C.15 Full pipeline scores with respect to substitutions generated by the Merriam generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.544	0.013	0.469
Syllable Count	0.577	0.013	0.435
Senses	0.833	0.008	0.176
Synonyms	0.833	0.013	0.180
Hypernyms	0.707	0.008	0.301
Hyponyms	0.649	0.008	0.360
Frequency: Simple Wiki	0.444	0.000	0.556
Frequency: Brown	0.766	0.013	0.247
Frequency: SUBTLEX	0.485	0.000	0.515
Frequency: SubIMDB	0.690	0.013	0.322
N-gram: Simple Wiki	0.912	0.004	0.092
N-gram: Brown	0.778	0.008	0.230
N-gram: SUBTLEX	0.833	0.013	0.180
N-gram: SubIMDB	0.778	0.004	0.226
Age of Acquisition	0.456	0.013	0.556
Familiarity	0.799	0.013	0.213
Concreteness	0.858	0.021	0.163
Imagery	0.833	0.021	0.188
Biran Ranker	0.603	0.021	0.418
Bott Ranker	0.778	0.013	0.234
Yamamoto Ranker	0.849	0.013	0.163
Horn Ranker	0.883	0.004	0.121
Glavas Ranker	0.900	0.004	0.105
Boundary Ranker	0.757	0.013	0.255

Table C.16 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.435	0.042	0.607
Syllable Count	0.485	0.038	0.552
Senses	0.678	0.025	0.347
Synonyms	0.686	0.025	0.339
Hypernyms	0.565	0.021	0.456
Hyponyms	0.548	0.025	0.477
Frequency: Simple Wiki	0.318	0.025	0.707
Frequency: Brown	0.686	0.025	0.339
Frequency: SUBTLEX	0.343	0.021	0.678
Frequency: SubIMDB	0.565	0.038	0.473
N-gram: Simple Wiki	0.824	0.017	0.192
N-gram: Brown	0.724	0.033	0.310
N-gram: SUBTLEX	0.732	0.038	0.305
N-gram: SubIMDB	0.749	0.033	0.285
Age of Acquisition	0.322	0.029	0.707
Familiarity	0.661	0.025	0.364
Concreteness	0.623	0.038	0.414
Imagery	0.661	0.025	0.364
Biran Ranker	0.456	0.050	0.594
Bott Ranker	0.669	0.025	0.356
Yamamoto Ranker	0.803	0.017	0.213
Horn Ranker	0.841	0.013	0.172
Glavas Ranker	0.845	0.025	0.180
Boundary Ranker	0.682	0.033	0.351

Table C.17 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.657	0.004	0.347
Syllable Count	0.686	0.004	0.318
Senses	0.916	0.000	0.084
Synonyms	0.916	0.000	0.084
Hypernyms	0.837	0.000	0.163
Hyponyms	0.774	0.000	0.226
Frequency: Simple Wiki	0.498	0.000	0.502
Frequency: Brown	0.787	0.004	0.218
Frequency: SUBTLEX	0.540	0.000	0.460
Frequency: SubIMDB	0.736	0.004	0.268
N-gram: Simple Wiki	0.933	0.000	0.067
N-gram: Brown	0.820	0.004	0.184
N-gram: SUBTLEX	0.887	0.008	0.121
N-gram: SubIMDB	0.833	0.000	0.167
Age of Acquisition	0.506	0.004	0.498
Familiarity	0.849	0.004	0.155
Concreteness	0.912	0.004	0.092
Imagery	0.887	0.004	0.117
Biran Ranker	0.703	0.008	0.305
Bott Ranker	0.879	0.004	0.126
Yamamoto Ranker	0.891	0.004	0.113
Horn Ranker	0.921	0.004	0.084
Glavas Ranker	0.916	0.000	0.084
Boundary Ranker	0.808	0.004	0.197

Table C.18 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.619	0.046	0.427
Syllable Count	0.640	0.054	0.414
Senses	0.628	0.029	0.402
Synonyms	0.603	0.042	0.439
Hypernyms	0.582	0.042	0.460
Hyponyms	0.598	0.042	0.444
Frequency: Simple Wiki	0.577	0.029	0.452
Frequency: Brown	0.611	0.025	0.414
Frequency: SUBTLEX	0.582	0.013	0.431
Frequency: SubIMDB	0.603	0.025	0.423
N-gram: Simple Wiki	0.791	0.046	0.255
N-gram: Brown	0.720	0.029	0.310
N-gram: SUBTLEX	0.728	0.059	0.331
N-gram: SubIMDB	0.736	0.054	0.318
Age of Acquisition	0.565	0.046	0.481
Familiarity	0.598	0.029	0.431
Concreteness	0.590	0.013	0.423
Imagery	0.590	0.025	0.435
Biran Ranker	0.611	0.046	0.435
Bott Ranker	0.644	0.033	0.389
Yamamoto Ranker	0.720	0.029	0.310
Horn Ranker	0.774	0.025	0.251
Glavas Ranker	0.766	0.038	0.272
Boundary Ranker	0.649	0.025	0.377

Table C.19 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.833	0.033	0.201
Syllable Count	0.854	0.042	0.188
Senses	0.858	0.029	0.172
Synonyms	0.874	0.046	0.172
Hypernyms	0.854	0.025	0.172
Hyponyms	0.854	0.025	0.172
Frequency: Simple Wiki	0.816	0.084	0.268
Frequency: Brown	0.921	0.008	0.088
Frequency: SUBTLEX	0.862	0.079	0.218
Frequency: SubIMDB	0.858	0.029	0.172
N-gram: Simple Wiki	0.904	0.025	0.121
N-gram: Brown	0.883	0.021	0.138
N-gram: SUBTLEX	0.912	0.025	0.113
N-gram: SubIMDB	0.912	0.021	0.109
Age of Acquisition	0.833	0.025	0.192
Familiarity	0.912	0.017	0.105
Concreteness	0.870	0.017	0.146
Imagery	0.908	0.050	0.142
Biran Ranker	0.828	0.033	0.205
Bott Ranker	0.858	0.025	0.167
Yamamoto Ranker	0.946	0.017	0.071
Horn Ranker	0.933	0.008	0.075
Glavas Ranker	0.946	0.021	0.075
Boundary Ranker	0.908	0.021	0.113

Table C.20 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.285	0.109	0.824
Syllable Count	0.339	0.130	0.791
Senses	0.297	0.075	0.778
Synonyms	0.230	0.084	0.854
Hypernyms	0.255	0.096	0.841
Hyponyms	0.222	0.084	0.862
Frequency: Simple Wiki	0.259	0.067	0.808
Frequency: Brown	0.285	0.084	0.799
Frequency: SUBTLEX	0.326	0.079	0.753
Frequency: SubIMDB	0.247	0.084	0.837
N-gram: Simple Wiki	0.577	0.100	0.523
N-gram: Brown	0.456	0.075	0.619
N-gram: SUBTLEX	0.469	0.113	0.644
N-gram: SubIMDB	0.477	0.113	0.636
Age of Acquisition	0.088	0.088	1.000
Familiarity	0.092	0.092	1.000
Concreteness	0.096	0.075	0.979
Imagery	0.105	0.092	0.987
Biran Ranker	0.280	0.121	0.841
Bott Ranker	0.331	0.100	0.770
Yamamoto Ranker	0.469	0.084	0.615
Horn Ranker	0.561	0.096	0.536
Glavas Ranker	0.552	0.088	0.536
Boundary Ranker	0.343	0.079	0.736

Table C.21 Full pipeline scores with respect to substitutions generated by the Merriam generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.410	0.134	0.724
Syllable Count	0.469	0.126	0.657
Senses	0.423	0.163	0.741
Synonyms	0.389	0.172	0.782
Hypernyms	0.381	0.138	0.757
Hyponyms	0.385	0.159	0.774
Frequency: Simple Wiki	0.393	0.025	0.632
Frequency: Brown	0.397	0.146	0.749
Frequency: SUBTLEX	0.431	0.021	0.590
Frequency: SubIMDB	0.322	0.172	0.849
N-gram: Simple Wiki	0.536	0.201	0.665
N-gram: Brown	0.523	0.188	0.665
N-gram: SUBTLEX	0.523	0.213	0.690
N-gram: SubIMDB	0.506	0.218	0.711
Age of Acquisition	0.151	0.151	1.000
Familiarity	0.155	0.155	1.000
Concreteness	0.121	0.121	1.000
Imagery	0.138	0.138	1.000
Biran Ranker	0.393	0.142	0.749
Bott Ranker	0.377	0.138	0.762
Yamamoto Ranker	0.510	0.109	0.598
Horn Ranker	0.364	0.172	0.808
Glavas Ranker	0.544	0.197	0.653
Boundary Ranker	0.377	0.192	0.816

Table C.22 Full pipeline scores with respect to substitutions generated by the Kauchak generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.640	0.013	0.372
Syllable Count	0.661	0.008	0.347
Senses	0.971	0.013	0.042
Synonyms	0.971	0.013	0.042
Hypernyms	0.849	0.013	0.163
Hyponyms	0.787	0.013	0.226
Frequency: Simple Wiki	0.460	0.004	0.544
Frequency: Brown	0.891	0.013	0.121
Frequency: SUBTLEX	0.460	0.004	0.544
Frequency: SubIMDB	0.770	0.013	0.243
N-gram: Simple Wiki	0.954	0.013	0.059
N-gram: Brown	0.891	0.008	0.117
N-gram: SUBTLEX	0.929	0.017	0.088
N-gram: SubIMDB	0.862	0.013	0.151
Age of Acquisition	0.490	0.004	0.515
Familiarity	0.946	0.013	0.067
Concreteness	0.870	0.013	0.142
Imagery	0.962	0.017	0.054
Biran Ranker	0.640	0.004	0.364
Bott Ranker	0.849	0.017	0.167
Yamamoto Ranker	0.929	0.013	0.084
Horn Ranker	0.912	0.013	0.100
Glavas Ranker	0.958	0.013	0.054
Boundary Ranker	0.812	0.013	0.201

Table C.23 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.586	0.021	0.435
Syllable Count	0.619	0.013	0.393
Senses	0.874	0.033	0.159
Synonyms	0.895	0.029	0.134
Hypernyms	0.828	0.021	0.192
Hyponyms	0.762	0.017	0.255
Frequency: Simple Wiki	0.418	0.008	0.590
Frequency: Brown	0.791	0.013	0.222
Frequency: SUBTLEX	0.423	0.004	0.582
Frequency: SubIMDB	0.665	0.013	0.347
N-gram: Simple Wiki	0.858	0.021	0.163
N-gram: Brown	0.841	0.017	0.176
N-gram: SUBTLEX	0.816	0.025	0.209
N-gram: SubIMDB	0.778	0.017	0.238
Age of Acquisition	0.435	0.013	0.577
Familiarity	0.837	0.008	0.172
Concreteness	0.757	0.021	0.264
Imagery	0.795	0.025	0.230
Biran Ranker	0.569	0.013	0.444
Bott Ranker	0.745	0.013	0.268
Yamamoto Ranker	0.854	0.008	0.155
Horn Ranker	0.803	0.021	0.218
Glavas Ranker	0.858	0.008	0.151
Boundary Ranker	0.732	0.017	0.285

Table C.24 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.661	0.000	0.339
Syllable Count	0.678	0.000	0.322
Senses	1.000	0.000	0.000
Synonyms	1.000	0.000	0.000
Hypernyms	0.887	0.000	0.113
Hyponyms	0.824	0.000	0.176
Frequency: Simple Wiki	0.423	0.004	0.582
Frequency: Brown	0.887	0.000	0.113
Frequency: SUBTLEX	0.423	0.004	0.582
Frequency: SubIMDB	0.795	0.000	0.205
N-gram: Simple Wiki	0.983	0.000	0.017
N-gram: Brown	0.891	0.000	0.109
N-gram: SUBTLEX	0.946	0.004	0.059
N-gram: SubIMDB	0.900	0.000	0.100
Age of Acquisition	0.519	0.000	0.481
Familiarity	1.000	0.000	0.000
Concreteness	0.904	0.000	0.096
Imagery	0.992	0.000	0.008
Biran Ranker	0.628	0.000	0.372
Bott Ranker	0.891	0.000	0.109
Yamamoto Ranker	0.971	0.000	0.029
Horn Ranker	0.954	0.000	0.046
Glavas Ranker	0.962	0.000	0.038
Boundary Ranker	0.858	0.000	0.142

Table C.25 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.682	0.067	0.385
Syllable Count	0.695	0.067	0.372
Senses	0.703	0.105	0.402
Synonyms	0.695	0.117	0.423
Hypernyms	0.686	0.084	0.397
Hyponyms	0.657	0.075	0.418
Frequency: Simple Wiki	0.695	0.029	0.335
Frequency: Brown	0.690	0.079	0.389
Frequency: SUBTLEX	0.690	0.013	0.322
Frequency: SubIMDB	0.632	0.088	0.456
N-gram: Simple Wiki	0.766	0.109	0.343
N-gram: Brown	0.741	0.092	0.351
N-gram: SUBTLEX	0.757	0.100	0.343
N-gram: SubIMDB	0.749	0.096	0.347
Age of Acquisition	0.611	0.059	0.448
Familiarity	0.674	0.079	0.406
Concreteness	0.644	0.050	0.406
Imagery	0.607	0.059	0.452
Biran Ranker	0.674	0.071	0.397
Bott Ranker	0.682	0.079	0.397
Yamamoto Ranker	0.745	0.054	0.310
Horn Ranker	0.690	0.092	0.402
Glavas Ranker	0.753	0.096	0.343
Boundary Ranker	0.678	0.092	0.414

Table C.26 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.879	0.075	0.197
Syllable Count	0.866	0.059	0.192
Senses	0.900	0.075	0.176
Synonyms	0.908	0.088	0.180
Hypernyms	0.887	0.054	0.167
Hyponyms	0.887	0.050	0.163
Frequency: Simple Wiki	0.937	0.029	0.092
Frequency: Brown	0.858	0.092	0.234
Frequency: SUBTLEX	0.900	0.013	0.113
Frequency: SubIMDB	0.845	0.113	0.268
N-gram: Simple Wiki	0.887	0.105	0.218
N-gram: Brown	0.912	0.071	0.159
N-gram: SUBTLEX	0.862	0.105	0.243
N-gram: SubIMDB	0.891	0.100	0.209
Age of Acquisition	0.862	0.100	0.238
Familiarity	0.858	0.100	0.243
Concreteness	0.929	0.063	0.134
Imagery	0.837	0.100	0.264
Biran Ranker	0.879	0.092	0.213
Bott Ranker	0.883	0.100	0.218
Yamamoto Ranker	0.933	0.092	0.159
Horn Ranker	0.883	0.109	0.226
Glavas Ranker	0.916	0.079	0.163
Boundary Ranker	0.870	0.113	0.243

Table C.27 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.343	0.109	0.766
Syllable Count	0.414	0.121	0.707
Senses	0.389	0.146	0.757
Synonyms	0.347	0.159	0.812
Hypernyms	0.351	0.130	0.778
Hyponyms	0.318	0.130	0.812
Frequency: Simple Wiki	0.427	0.050	0.623
Frequency: Brown	0.343	0.117	0.774
Frequency: SUBTLEX	0.460	0.063	0.603
Frequency: SubIMDB	0.255	0.130	0.874
N-gram: Simple Wiki	0.498	0.188	0.690
N-gram: Brown	0.444	0.151	0.707
N-gram: SUBTLEX	0.460	0.188	0.728
N-gram: SubIMDB	0.435	0.184	0.749
Age of Acquisition	0.105	0.100	0.996
Familiarity	0.126	0.126	1.000
Concreteness	0.126	0.126	1.000
Imagery	0.142	0.138	0.996
Biran Ranker	0.318	0.105	0.787
Bott Ranker	0.326	0.113	0.787
Yamamoto Ranker	0.435	0.071	0.636
Horn Ranker	0.335	0.155	0.820
Glavas Ranker	0.473	0.159	0.686
Boundary Ranker	0.318	0.151	0.833

Table C.28 Full pipeline scores with respect to substitutions generated by the Kauchak generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.314	0.126	0.812
Syllable Count	0.297	0.134	0.837
Senses	0.310	0.105	0.795
Synonyms	0.255	0.113	0.858
Hypernyms	0.351	0.121	0.770
Hyponyms	0.335	0.146	0.812
Frequency: Simple Wiki	0.197	0.063	0.866
Frequency: Brown	0.335	0.117	0.782
Frequency: SUBTLEX	0.247	0.067	0.820
Frequency: SubIMDB	0.276	0.151	0.874
N-gram: Simple Wiki	0.569	0.151	0.582
N-gram: Brown	0.456	0.142	0.686
N-gram: SUBTLEX	0.498	0.146	0.649
N-gram: SubIMDB	0.490	0.146	0.657
Age of Acquisition	0.067	0.067	1.000
Familiarity	0.172	0.172	1.000
Concreteness	0.134	0.134	1.000
Imagery	0.146	0.146	1.000
Biran Ranker	0.314	0.126	0.812
Bott Ranker	0.360	0.126	0.766
Yamamoto Ranker	0.519	0.100	0.582
Horn Ranker	0.552	0.146	0.594
Glavas Ranker	0.594	0.126	0.531
Boundary Ranker	0.381	0.159	0.778

Table C.29 Full pipeline scores with respect to substitutions generated by the WordNet generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.699	0.025	0.326
Syllable Count	0.741	0.025	0.285
Senses	0.837	0.029	0.192
Synonyms	0.833	0.038	0.205
Hypernyms	0.791	0.033	0.243
Hyponyms	0.782	0.033	0.251
Frequency: Simple Wiki	0.540	0.000	0.460
Frequency: Brown	0.849	0.025	0.176
Frequency: SUBTLEX	0.594	0.000	0.406
Frequency: SubIMDB	0.803	0.029	0.226
N-gram: Simple Wiki	0.941	0.029	0.088
N-gram: Brown	0.841	0.021	0.180
N-gram: SUBTLEX	0.916	0.033	0.117
N-gram: SubIMDB	0.858	0.029	0.172
Age of Acquisition	0.594	0.025	0.431
Familiarity	0.891	0.029	0.138
Concreteness	0.854	0.042	0.188
Imagery	0.858	0.042	0.184
Biran Ranker	0.728	0.029	0.301
Bott Ranker	0.849	0.029	0.180
Yamamoto Ranker	0.912	0.029	0.117
Horn Ranker	0.937	0.013	0.075
Glavas Ranker	0.946	0.025	0.079
Boundary Ranker	0.841	0.029	0.188

Table C.30 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.611	0.038	0.427
Syllable Count	0.644	0.038	0.393
Senses	0.782	0.029	0.247
Synonyms	0.749	0.033	0.285
Hypernyms	0.749	0.029	0.280
Hyponyms	0.724	0.033	0.310
Frequency: Simple Wiki	0.372	0.021	0.649
Frequency: Brown	0.745	0.029	0.285
Frequency: SUBTLEX	0.431	0.025	0.594
Frequency: SubIMDB	0.686	0.042	0.356
N-gram: Simple Wiki	0.874	0.025	0.151
N-gram: Brown	0.736	0.025	0.289
N-gram: SUBTLEX	0.837	0.050	0.213
N-gram: SubIMDB	0.791	0.042	0.251
Age of Acquisition	0.460	0.017	0.556
Familiarity	0.782	0.042	0.259
Concreteness	0.695	0.033	0.339
Imagery	0.745	0.033	0.289
Biran Ranker	0.636	0.038	0.402
Bott Ranker	0.787	0.038	0.251
Yamamoto Ranker	0.904	0.021	0.117
Horn Ranker	0.895	0.017	0.121
Glavas Ranker	0.887	0.029	0.142
Boundary Ranker	0.774	0.046	0.272

Table C.31 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.866	0.004	0.138
Syllable Count	0.900	0.004	0.105
Senses	0.975	0.000	0.025
Synonyms	0.975	0.000	0.025
Hypernyms	0.975	0.000	0.025
Hyponyms	0.967	0.000	0.033
Frequency: Simple Wiki	0.803	0.000	0.197
Frequency: Brown	0.941	0.000	0.059
Frequency: SUBTLEX	0.812	0.000	0.188
Frequency: SubIMDB	0.904	0.000	0.096
N-gram: Simple Wiki	0.987	0.000	0.013
N-gram: Brown	0.946	0.000	0.054
N-gram: SUBTLEX	0.987	0.004	0.017
N-gram: SubIMDB	0.929	0.000	0.071
Age of Acquisition	0.812	0.000	0.188
Familiarity	1.000	0.000	0.000
Concreteness	0.992	0.000	0.008
Imagery	0.996	0.000	0.004
Biran Ranker	0.879	0.004	0.126
Bott Ranker	0.975	0.000	0.025
Yamamoto Ranker	0.987	0.000	0.013
Horn Ranker	0.987	0.000	0.013
Glavas Ranker	0.971	0.000	0.029
Boundary Ranker	0.925	0.000	0.075

Table C.32 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.682	0.079	0.397
Syllable Count	0.674	0.088	0.414
Senses	0.669	0.063	0.393
Synonyms	0.636	0.071	0.435
Hypernyms	0.690	0.067	0.377
Hyponyms	0.669	0.088	0.418
Frequency: Simple Wiki	0.590	0.042	0.452
Frequency: Brown	0.690	0.071	0.381
Frequency: SUBTLEX	0.644	0.033	0.389
Frequency: SubIMDB	0.653	0.084	0.431
N-gram: Simple Wiki	0.778	0.059	0.280
N-gram: Brown	0.732	0.084	0.351
N-gram: SUBTLEX	0.753	0.084	0.331
N-gram: SubIMDB	0.757	0.071	0.314
Age of Acquisition	0.615	0.054	0.439
Familiarity	0.665	0.079	0.414
Concreteness	0.678	0.054	0.377
Imagery	0.686	0.059	0.372
Biran Ranker	0.661	0.067	0.406
Bott Ranker	0.690	0.067	0.377
Yamamoto Ranker	0.770	0.042	0.272
Horn Ranker	0.770	0.071	0.301
Glavas Ranker	0.820	0.059	0.238
Boundary Ranker	0.715	0.084	0.368

Table C.33 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.879	0.050	0.172
Syllable Count	0.891	0.050	0.159
Senses	0.891	0.046	0.155
Synonyms	0.900	0.050	0.151
Hypernyms	0.866	0.067	0.201
Hyponyms	0.900	0.075	0.176
Frequency: Simple Wiki	0.904	0.067	0.163
Frequency: Brown	0.900	0.050	0.151
Frequency: SUBTLEX	0.912	0.063	0.151
Frequency: SubIMDB	0.883	0.059	0.176
N-gram: Simple Wiki	0.891	0.046	0.155
N-gram: Brown	0.904	0.046	0.142
N-gram: SUBTLEX	0.916	0.063	0.146
N-gram: SubIMDB	0.916	0.050	0.134
Age of Acquisition	0.874	0.050	0.176
Familiarity	0.887	0.050	0.163
Concreteness	0.891	0.042	0.151
Imagery	0.883	0.042	0.159
Biran Ranker	0.879	0.054	0.176
Bott Ranker	0.895	0.050	0.155
Yamamoto Ranker	0.912	0.038	0.126
Horn Ranker	0.925	0.046	0.121
Glavas Ranker	0.954	0.038	0.084
Boundary Ranker	0.908	0.054	0.146

Table C.34 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.318	0.130	0.812
Syllable Count	0.310	0.142	0.833
Senses	0.339	0.109	0.770
Synonyms	0.280	0.117	0.837
Hypernyms	0.381	0.134	0.753
Hyponyms	0.368	0.163	0.795
Frequency: Simple Wiki	0.247	0.092	0.845
Frequency: Brown	0.347	0.126	0.778
Frequency: SUBTLEX	0.289	0.088	0.799
Frequency: SubIMDB	0.289	0.159	0.870
N-gram: Simple Wiki	0.569	0.151	0.582
N-gram: Brown	0.456	0.142	0.686
N-gram: SUBTLEX	0.498	0.146	0.649
N-gram: SubIMDB	0.494	0.151	0.657
Age of Acquisition	0.092	0.092	1.000
Familiarity	0.176	0.176	1.000
Concreteness	0.159	0.159	1.000
Imagery	0.159	0.159	1.000
Biran Ranker	0.326	0.138	0.812
Bott Ranker	0.377	0.142	0.766
Yamamoto Ranker	0.531	0.105	0.573
Horn Ranker	0.561	0.142	0.582
Glavas Ranker	0.598	0.126	0.527
Boundary Ranker	0.389	0.163	0.774

Table C.35 Full pipeline scores with respect to substitutions generated by the WordNet generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.293	0.021	0.728
Syllable Count	0.301	0.029	0.728
Senses	0.343	0.033	0.690
Synonyms	0.339	0.025	0.686
Hypernyms	0.293	0.033	0.741
Hyponyms	0.276	0.013	0.736
Frequency: Simple Wiki	0.423	0.017	0.594
Frequency: Brown	0.339	0.029	0.690
Frequency: SUBTLEX	0.435	0.000	0.565
Frequency: SubIMDB	0.276	0.017	0.741
N-gram: Simple Wiki	0.556	0.079	0.523
N-gram: Brown	0.418	0.067	0.649
N-gram: SUBTLEX	0.431	0.038	0.607
N-gram: SubIMDB	0.481	0.050	0.569
Age of Acquisition	0.017	0.017	1.000
Familiarity	0.021	0.021	1.000
Concreteness	0.046	0.046	1.000
Imagery	0.033	0.033	1.000
Biran Ranker	0.289	0.025	0.736
Bott Ranker	0.301	0.038	0.736
Yamamoto Ranker	0.444	0.025	0.582
Horn Ranker	0.435	0.046	0.611
Glavas Ranker	0.515	0.059	0.544
Boundary Ranker	0.326	0.038	0.711

Table C.36 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.866	0.000	0.134
Syllable Count	0.874	0.000	0.126
Senses	0.954	0.000	0.046
Synonyms	0.958	0.000	0.042
Hypernyms	0.950	0.000	0.050
Hyponyms	0.946	0.000	0.054
Frequency: Simple Wiki	0.862	0.000	0.138
Frequency: Brown	0.820	0.000	0.180
Frequency: SUBTLEX	0.900	0.000	0.100
Frequency: SubIMDB	0.795	0.000	0.205
N-gram: Simple Wiki	0.921	0.000	0.079
N-gram: Brown	0.862	0.000	0.138
N-gram: SUBTLEX	0.891	0.000	0.109
N-gram: SubIMDB	0.895	0.000	0.105
Age of Acquisition	0.795	0.000	0.205
Familiarity	0.921	0.000	0.079
Concreteness	0.950	0.004	0.054
Imagery	0.954	0.004	0.050
Biran Ranker	0.870	0.000	0.130
Bott Ranker	0.862	0.000	0.138
Yamamoto Ranker	0.849	0.000	0.151
Horn Ranker	0.908	0.000	0.092
Glavas Ranker	0.883	0.000	0.117
Boundary Ranker	0.824	0.000	0.176

Table C.37 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.799	0.000	0.201
Syllable Count	0.808	0.000	0.192
Senses	0.929	0.000	0.071
Synonyms	0.916	0.000	0.084
Hypernyms	0.895	0.000	0.105
Hyponyms	0.849	0.000	0.151
Frequency: Simple Wiki	0.762	0.004	0.243
Frequency: Brown	0.774	0.000	0.226
Frequency: SUBTLEX	0.808	0.000	0.192
Frequency: SubIMDB	0.736	0.000	0.264
N-gram: Simple Wiki	0.874	0.004	0.130
N-gram: Brown	0.808	0.004	0.197
N-gram: SUBTLEX	0.837	0.000	0.163
N-gram: SubIMDB	0.841	0.000	0.159
Age of Acquisition	0.715	0.000	0.285
Familiarity	0.849	0.008	0.159
Concreteness	0.824	0.004	0.180
Imagery	0.820	0.000	0.180
Biran Ranker	0.795	0.000	0.205
Bott Ranker	0.816	0.000	0.184
Yamamoto Ranker	0.837	0.000	0.163
Horn Ranker	0.895	0.004	0.109
Glavas Ranker	0.828	0.000	0.172
Boundary Ranker	0.770	0.000	0.230

Table C.38 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.900	0.000	0.100
Syllable Count	0.904	0.000	0.096
Senses	0.987	0.000	0.013
Synonyms	0.987	0.000	0.013
Hypernyms	0.975	0.000	0.025
Hyponyms	0.971	0.000	0.029
Frequency: Simple Wiki	0.900	0.000	0.100
Frequency: Brown	0.837	0.000	0.163
Frequency: SUBTLEX	0.916	0.000	0.084
Frequency: SubIMDB	0.816	0.000	0.184
N-gram: Simple Wiki	0.933	0.000	0.067
N-gram: Brown	0.874	0.000	0.126
N-gram: SUBTLEX	0.904	0.000	0.096
N-gram: SubIMDB	0.912	0.000	0.088
Age of Acquisition	0.812	0.000	0.188
Familiarity	0.971	0.000	0.029
Concreteness	0.979	0.000	0.021
Imagery	0.983	0.000	0.017
Biran Ranker	0.891	0.000	0.109
Bott Ranker	0.874	0.000	0.126
Yamamoto Ranker	0.866	0.000	0.134
Horn Ranker	0.937	0.000	0.063
Glavas Ranker	0.874	0.000	0.126
Boundary Ranker	0.837	0.000	0.163

Table C.39 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.611	0.017	0.406
Syllable Count	0.615	0.021	0.406
Senses	0.632	0.033	0.402
Synonyms	0.632	0.029	0.397
Hypernyms	0.611	0.033	0.423
Hyponyms	0.598	0.021	0.423
Frequency: Simple Wiki	0.715	0.008	0.293
Frequency: Brown	0.619	0.025	0.406
Frequency: SUBTLEX	0.720	0.000	0.280
Frequency: SubIMDB	0.594	0.017	0.423
N-gram: Simple Wiki	0.762	0.046	0.285
N-gram: Brown	0.678	0.038	0.360
N-gram: SUBTLEX	0.707	0.033	0.326
N-gram: SubIMDB	0.753	0.033	0.280
Age of Acquisition	0.594	0.017	0.423
Familiarity	0.619	0.021	0.402
Concreteness	0.640	0.025	0.385
Imagery	0.632	0.025	0.393
Biran Ranker	0.611	0.017	0.406
Bott Ranker	0.607	0.029	0.423
Yamamoto Ranker	0.707	0.017	0.310
Horn Ranker	0.711	0.038	0.326
Glavas Ranker	0.728	0.038	0.310
Boundary Ranker	0.623	0.033	0.410

Table C.40 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.946	0.021	0.075
Syllable Count	0.946	0.029	0.084
Senses	0.979	0.029	0.050
Synonyms	0.983	0.033	0.050
Hypernyms	0.971	0.029	0.059
Hyponyms	0.958	0.033	0.075
Frequency: Simple Wiki	0.946	0.008	0.063
Frequency: Brown	0.958	0.021	0.063
Frequency: SUBTLEX	0.958	0.004	0.046
Frequency: SubIMDB	0.950	0.029	0.079
N-gram: Simple Wiki	0.979	0.029	0.050
N-gram: Brown	0.962	0.025	0.063
N-gram: SUBTLEX	0.962	0.029	0.067
N-gram: SubIMDB	0.962	0.025	0.063
Age of Acquisition	0.937	0.021	0.084
Familiarity	0.937	0.029	0.092
Concreteness	0.941	0.021	0.079
Imagery	0.941	0.025	0.084
Biran Ranker	0.941	0.025	0.084
Bott Ranker	0.950	0.021	0.071
Yamamoto Ranker	0.992	0.004	0.013
Horn Ranker	0.979	0.021	0.042
Glavas Ranker	0.971	0.021	0.050
Boundary Ranker	0.954	0.029	0.075

Table C.41 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.305	0.025	0.720
Syllable Count	0.335	0.046	0.711
Senses	0.356	0.033	0.678
Synonyms	0.360	0.025	0.665
Hypernyms	0.297	0.033	0.736
Hyponyms	0.297	0.029	0.732
Frequency: Simple Wiki	0.582	0.046	0.464
Frequency: Brown	0.351	0.042	0.690
Frequency: SUBTLEX	0.653	0.025	0.372
Frequency: SubIMDB	0.289	0.029	0.741
N-gram: Simple Wiki	0.561	0.079	0.519
N-gram: Brown	0.435	0.084	0.649
N-gram: SUBTLEX	0.444	0.046	0.603
N-gram: SubIMDB	0.485	0.050	0.565
Age of Acquisition	0.029	0.029	1.000
Familiarity	0.033	0.033	1.000
Concreteness	0.075	0.075	1.000
Imagery	0.079	0.075	0.996
Biran Ranker	0.289	0.025	0.736
Bott Ranker	0.301	0.038	0.736
Yamamoto Ranker	0.506	0.025	0.519
Horn Ranker	0.452	0.050	0.598
Glavas Ranker	0.544	0.059	0.515
Boundary Ranker	0.339	0.050	0.711

Table C.42 Full pipeline scores with respect to substitutions generated by the Yamamoto generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.209	0.126	0.916
Syllable Count	0.180	0.117	0.937
Senses	0.280	0.130	0.849
Synonyms	0.372	0.138	0.766
Hypernyms	0.264	0.142	0.879
Hyponyms	0.293	0.142	0.849
Frequency: Simple Wiki	0.042	0.029	0.987
Frequency: Brown	0.293	0.192	0.900
Frequency: SUBTLEX	0.017	0.008	0.992
Frequency: SubIMDB	0.310	0.230	0.921
N-gram: Simple Wiki	0.494	0.192	0.699
N-gram: Brown	0.414	0.172	0.757
N-gram: SUBTLEX	0.481	0.201	0.720
N-gram: SubIMDB	0.494	0.213	0.720
Age of Acquisition	0.050	0.050	1.000
Familiarity	0.230	0.230	1.000
Concreteness	0.218	0.218	1.000
Imagery	0.247	0.247	1.000
Biran Ranker	0.259	0.155	0.895
Bott Ranker	0.393	0.163	0.770
Yamamoto Ranker	0.510	0.134	0.623
Horn Ranker	0.552	0.192	0.640
Glavas Ranker	0.590	0.163	0.573
Boundary Ranker	0.418	0.218	0.799

Table C.43 Full pipeline scores with respect to substitutions generated by the Glavas generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.770	0.029	0.259
Syllable Count	0.657	0.029	0.372
Senses	0.933	0.017	0.084
Synonyms	0.929	0.017	0.088
Hypernyms	0.787	0.021	0.234
Hyponyms	0.770	0.021	0.251
Frequency: Simple Wiki	0.356	0.008	0.653
Frequency: Brown	0.833	0.021	0.188
Frequency: SUBTLEX	0.360	0.013	0.653
Frequency: SubIMDB	0.753	0.021	0.268
N-gram: Simple Wiki	0.950	0.025	0.075
N-gram: Brown	0.803	0.013	0.209
N-gram: SUBTLEX	0.925	0.025	0.100
N-gram: SubIMDB	0.854	0.017	0.163
Age of Acquisition	0.377	0.029	0.653
Familiarity	0.967	0.017	0.050
Concreteness	0.975	0.025	0.050
Imagery	0.979	0.025	0.046
Biran Ranker	0.774	0.033	0.259
Bott Ranker	0.891	0.029	0.138
Yamamoto Ranker	0.950	0.021	0.071
Horn Ranker	0.954	0.013	0.059
Glavas Ranker	0.941	0.021	0.079
Boundary Ranker	0.808	0.021	0.213

Table C.44 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.603	0.046	0.444
Syllable Count	0.510	0.050	0.540
Senses	0.707	0.029	0.322
Synonyms	0.732	0.025	0.293
Hypernyms	0.636	0.042	0.406
Hyponyms	0.623	0.063	0.439
Frequency: Simple Wiki	0.126	0.017	0.891
Frequency: Brown	0.649	0.050	0.402
Frequency: SUBTLEX	0.121	0.017	0.895
Frequency: SubIMDB	0.548	0.063	0.515
N-gram: Simple Wiki	0.816	0.046	0.230
N-gram: Brown	0.657	0.038	0.381
N-gram: SUBTLEX	0.762	0.054	0.293
N-gram: SubIMDB	0.703	0.042	0.339
Age of Acquisition	0.222	0.046	0.824
Familiarity	0.766	0.050	0.285
Concreteness	0.674	0.054	0.381
Imagery	0.699	0.050	0.351
Biran Ranker	0.577	0.046	0.469
Bott Ranker	0.724	0.054	0.331
Yamamoto Ranker	0.837	0.033	0.197
Horn Ranker	0.837	0.042	0.205
Glavas Ranker	0.824	0.038	0.213
Boundary Ranker	0.653	0.059	0.406

Table C.45 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.837	0.000	0.163
Syllable Count	0.728	0.004	0.276
Senses	0.992	0.000	0.008
Synonyms	0.992	0.000	0.008
Hypernyms	0.854	0.000	0.146
Hyponyms	0.837	0.000	0.163
Frequency: Simple Wiki	0.439	0.004	0.565
Frequency: Brown	0.833	0.000	0.167
Frequency: SUBTLEX	0.435	0.004	0.569
Frequency: SubIMDB	0.778	0.000	0.222
N-gram: Simple Wiki	0.971	0.004	0.033
N-gram: Brown	0.828	0.000	0.172
N-gram: SUBTLEX	0.925	0.008	0.084
N-gram: SubIMDB	0.866	0.000	0.134
Age of Acquisition	0.448	0.004	0.556
Familiarity	0.987	0.000	0.013
Concreteness	1.000	0.000	0.000
Imagery	1.000	0.000	0.000
Biran Ranker	0.799	0.004	0.205
Bott Ranker	0.954	0.004	0.050
Yamamoto Ranker	0.967	0.000	0.033
Horn Ranker	0.967	0.004	0.038
Glavas Ranker	0.941	0.000	0.059
Boundary Ranker	0.828	0.000	0.172

Table C.46 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.669	0.067	0.397
Syllable Count	0.674	0.067	0.393
Senses	0.665	0.071	0.406
Synonyms	0.707	0.075	0.368
Hypernyms	0.653	0.079	0.427
Hyponyms	0.665	0.075	0.410
Frequency: Simple Wiki	0.469	0.017	0.548
Frequency: Brown	0.669	0.113	0.444
Frequency: SUBTLEX	0.477	0.017	0.540
Frequency: SubIMDB	0.690	0.146	0.456
N-gram: Simple Wiki	0.778	0.084	0.305
N-gram: Brown	0.720	0.088	0.368
N-gram: SUBTLEX	0.766	0.100	0.335
N-gram: SubIMDB	0.778	0.134	0.356
Age of Acquisition	0.527	0.067	0.540
Familiarity	0.695	0.138	0.444
Concreteness	0.653	0.088	0.435
Imagery	0.682	0.092	0.410
Biran Ranker	0.695	0.084	0.389
Bott Ranker	0.728	0.113	0.385
Yamamoto Ranker	0.799	0.084	0.285
Horn Ranker	0.816	0.109	0.293
Glavas Ranker	0.854	0.096	0.243
Boundary Ranker	0.745	0.126	0.381

Table C.47 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.812	0.059	0.247
Syllable Count	0.816	0.088	0.272
Senses	0.774	0.079	0.305
Synonyms	0.736	0.050	0.314
Hypernyms	0.703	0.079	0.377
Hyponyms	0.757	0.071	0.314
Frequency: Simple Wiki	0.678	0.096	0.418
Frequency: Brown	0.820	0.059	0.238
Frequency: SUBTLEX	0.715	0.130	0.414
Frequency: SubIMDB	0.762	0.071	0.310
N-gram: Simple Wiki	0.808	0.067	0.259
N-gram: Brown	0.841	0.071	0.230
N-gram: SUBTLEX	0.791	0.071	0.280
N-gram: SubIMDB	0.808	0.059	0.251
Age of Acquisition	0.732	0.067	0.335
Familiarity	0.812	0.071	0.259
Concreteness	0.816	0.067	0.251
Imagery	0.816	0.071	0.255
Biran Ranker	0.812	0.071	0.259
Bott Ranker	0.787	0.079	0.293
Yamamoto Ranker	0.874	0.067	0.192
Horn Ranker	0.858	0.075	0.218
Glavas Ranker	0.866	0.046	0.180
Boundary Ranker	0.787	0.067	0.280

Table C.48 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.385	0.184	0.799
Syllable Count	0.322	0.126	0.803
Senses	0.335	0.113	0.778
Synonyms	0.427	0.142	0.715
Hypernyms	0.377	0.184	0.808
Hyponyms	0.343	0.138	0.795
Frequency: Simple Wiki	0.063	0.033	0.971
Frequency: Brown	0.364	0.234	0.870
Frequency: SUBTLEX	0.067	0.042	0.975
Frequency: SubIMDB	0.339	0.255	0.916
N-gram: Simple Wiki	0.510	0.176	0.665
N-gram: Brown	0.464	0.176	0.711
N-gram: SUBTLEX	0.498	0.192	0.695
N-gram: SubIMDB	0.519	0.226	0.707
Age of Acquisition	0.088	0.088	1.000
Familiarity	0.255	0.255	1.000
Concreteness	0.159	0.159	1.000
Imagery	0.276	0.276	1.000
Biran Ranker	0.389	0.180	0.791
Bott Ranker	0.444	0.213	0.770
Yamamoto Ranker	0.598	0.167	0.569
Horn Ranker	0.590	0.209	0.619
Glavas Ranker	0.640	0.176	0.536
Boundary Ranker	0.456	0.234	0.778

Table C.49 Full pipeline scores with respect to substitutions generated by the Glavas generator, as selected by the Unsupervised Boundary Ranking Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.234	0.100	0.866
Syllable Count	0.326	0.142	0.816
Senses	0.259	0.088	0.828
Synonyms	0.259	0.092	0.833
Hypernyms	0.234	0.088	0.854
Hyponyms	0.243	0.092	0.849
Frequency: Simple Wiki	0.247	0.059	0.812
Frequency: Brown	0.243	0.059	0.816
Frequency: SUBTLEX	0.414	0.054	0.640
Frequency: SubIMDB	0.218	0.063	0.845
N-gram: Simple Wiki	0.527	0.121	0.594
N-gram: Brown	0.372	0.096	0.724
N-gram: SUBTLEX	0.456	0.117	0.661
N-gram: SubIMDB	0.477	0.138	0.661
Age of Acquisition	0.113	0.113	1.000
Familiarity	0.100	0.100	1.000
Concreteness	0.113	0.113	1.000
Imagery	0.113	0.113	1.000
Biran Ranker	0.234	0.084	0.849
Bott Ranker	0.247	0.084	0.837
Yamamoto Ranker	0.452	0.054	0.603
Horn Ranker	0.448	0.105	0.657
Glavas Ranker	0.523	0.092	0.569
Boundary Ranker	0.305	0.092	0.787

Table C.50 Full pipeline scores with respect to substitutions generated by the Biran generator, without any selection

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.686	0.025	0.339
Syllable Count	0.749	0.038	0.289
Senses	0.883	0.025	0.142
Synonyms	0.879	0.029	0.151
Hypernyms	0.879	0.025	0.146
Hyponyms	0.866	0.025	0.159
Frequency: Simple Wiki	0.682	0.013	0.331
Frequency: Brown	0.904	0.021	0.117
Frequency: SUBTLEX	0.728	0.013	0.285
Frequency: SubIMDB	0.870	0.025	0.155
N-gram: Simple Wiki	0.958	0.021	0.063
N-gram: Brown	0.912	0.013	0.100
N-gram: SUBTLEX	0.937	0.025	0.088
N-gram: SubIMDB	0.891	0.025	0.134
Age of Acquisition	0.695	0.025	0.331
Familiarity	0.921	0.025	0.105
Concreteness	0.916	0.033	0.117
Imagery	0.925	0.033	0.109
Biran Ranker	0.720	0.025	0.305
Bott Ranker	0.854	0.025	0.172
Yamamoto Ranker	0.933	0.025	0.092
Horn Ranker	0.954	0.013	0.059
Glavas Ranker	0.967	0.021	0.054
Boundary Ranker	0.891	0.025	0.134

Table C.51 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the First Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.582	0.038	0.456
Syllable Count	0.699	0.054	0.356
Senses	0.837	0.025	0.188
Synonyms	0.820	0.021	0.201
Hypernyms	0.849	0.017	0.167
Hyponyms	0.824	0.017	0.192
Frequency: Simple Wiki	0.640	0.033	0.393
Frequency: Brown	0.841	0.033	0.192
Frequency: SUBTLEX	0.669	0.029	0.360
Frequency: SubIMDB	0.812	0.042	0.230
N-gram: Simple Wiki	0.904	0.029	0.126
N-gram: Brown	0.874	0.038	0.163
N-gram: SUBTLEX	0.904	0.054	0.151
N-gram: SubIMDB	0.874	0.042	0.167
Age of Acquisition	0.615	0.042	0.427
Familiarity	0.841	0.029	0.188
Concreteness	0.799	0.033	0.234
Imagery	0.808	0.029	0.222
Biran Ranker	0.619	0.042	0.423
Bott Ranker	0.787	0.038	0.251
Yamamoto Ranker	0.916	0.013	0.096
Horn Ranker	0.921	0.013	0.092
Glavas Ranker	0.921	0.013	0.092
Boundary Ranker	0.854	0.038	0.184

Table C.52 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the Lesk Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.833	0.004	0.172
Syllable Count	0.866	0.004	0.138
Senses	0.987	0.000	0.013
Synonyms	0.987	0.000	0.013
Hypernyms	0.987	0.000	0.013
Hyponyms	0.975	0.000	0.025
Frequency: Simple Wiki	0.828	0.004	0.176
Frequency: Brown	0.958	0.000	0.042
Frequency: SUBTLEX	0.837	0.004	0.167
Frequency: SubIMDB	0.950	0.000	0.050
N-gram: Simple Wiki	0.996	0.000	0.004
N-gram: Brown	0.971	0.000	0.029
N-gram: SUBTLEX	0.992	0.000	0.008
N-gram: SubIMDB	0.958	0.000	0.042
Age of Acquisition	0.833	0.004	0.172
Familiarity	1.000	0.000	0.000
Concreteness	1.000	0.000	0.000
Imagery	1.000	0.000	0.000
Biran Ranker	0.858	0.004	0.146
Bott Ranker	0.971	0.000	0.029
Yamamoto Ranker	0.992	0.000	0.008
Horn Ranker	0.996	0.000	0.004
Glavas Ranker	0.987	0.000	0.013
Boundary Ranker	0.962	0.000	0.038

Table C.53 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the Path Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.628	0.071	0.444
Syllable Count	0.690	0.088	0.397
Senses	0.644	0.063	0.418
Synonyms	0.649	0.071	0.423
Hypernyms	0.653	0.071	0.418
Hyponyms	0.628	0.059	0.431
Frequency: Simple Wiki	0.678	0.038	0.360
Frequency: Brown	0.628	0.050	0.423
Frequency: SUBTLEX	0.757	0.025	0.268
Frequency: SubIMDB	0.628	0.059	0.431
N-gram: Simple Wiki	0.766	0.059	0.293
N-gram: Brown	0.703	0.071	0.368
N-gram: SUBTLEX	0.757	0.079	0.322
N-gram: SubIMDB	0.770	0.071	0.301
Age of Acquisition	0.632	0.071	0.439
Familiarity	0.632	0.054	0.423
Concreteness	0.674	0.046	0.372
Imagery	0.653	0.050	0.397
Biran Ranker	0.644	0.079	0.435
Bott Ranker	0.632	0.063	0.431
Yamamoto Ranker	0.736	0.029	0.293
Horn Ranker	0.724	0.059	0.335
Glavas Ranker	0.791	0.054	0.264
Boundary Ranker	0.665	0.067	0.402

Table C.54 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the Biran Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.870	0.063	0.192
Syllable Count	0.895	0.063	0.167
Senses	0.916	0.059	0.142
Synonyms	0.908	0.059	0.151
Hypernyms	0.891	0.059	0.167
Hyponyms	0.925	0.063	0.138
Frequency: Simple Wiki	0.925	0.013	0.088
Frequency: Brown	0.941	0.050	0.109
Frequency: SUBTLEX	0.925	0.000	0.075
Frequency: SubIMDB	0.929	0.063	0.134
N-gram: Simple Wiki	0.921	0.046	0.126
N-gram: Brown	0.933	0.042	0.109
N-gram: SUBTLEX	0.958	0.054	0.096
N-gram: SubIMDB	0.946	0.050	0.105
Age of Acquisition	0.887	0.054	0.167
Familiarity	0.929	0.050	0.121
Concreteness	0.887	0.038	0.151
Imagery	0.937	0.046	0.109
Biran Ranker	0.866	0.059	0.192
Bott Ranker	0.891	0.050	0.159
Yamamoto Ranker	0.912	0.029	0.117
Horn Ranker	0.950	0.033	0.084
Glavas Ranker	0.967	0.046	0.079
Boundary Ranker	0.937	0.059	0.121

Table C.55 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the Belder Selector

Ranker	Precision	Accuracy	Changed Proportion
Word Length	0.259	0.126	0.866
Syllable Count	0.339	0.155	0.816
Senses	0.272	0.088	0.816
Synonyms	0.272	0.096	0.824
Hypernyms	0.243	0.088	0.845
Hyponyms	0.259	0.100	0.841
Frequency: Simple Wiki	0.310	0.079	0.770
Frequency: Brown	0.259	0.071	0.812
Frequency: SUBTLEX	0.531	0.071	0.540
Frequency: SubIMDB	0.234	0.079	0.845
N-gram: Simple Wiki	0.527	0.121	0.594
N-gram: Brown	0.377	0.096	0.720
N-gram: SUBTLEX	0.460	0.117	0.657
N-gram: SubIMDB	0.481	0.138	0.657
Age of Acquisition	0.130	0.130	1.000
Familiarity	0.109	0.109	1.000
Concreteness	0.121	0.121	1.000
Imagery	0.121	0.121	1.000
Biran Ranker	0.255	0.105	0.849
Bott Ranker	0.259	0.096	0.837
Yamamoto Ranker	0.473	0.063	0.590
Horn Ranker	0.456	0.109	0.653
Glavas Ranker	0.527	0.092	0.565
Boundary Ranker	0.318	0.100	0.782

Table C.56 Full pipeline scores with respect to substitutions generated by the Biran generator, as selected by the Unsupervised Boundary Ranking Selector