BEYOND TOPIC-BASED REPRESENTATIONS FOR TEXT MINING

BY

SEAN ALEXANDER MASSUNG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor ChengXiang Zhai, Chair
Associate Professor Julia Hockenmaier
Professor Dan Roth
Professor Steve LaValle
Associate Professor Qiaozhu Mei, University of Michigan

ABSTRACT

A massive amount of online information is natural language text: newspapers, blog articles, forum posts and comments, tweets, scientific literature, government documents, and more. While in general, all kinds of online information is useful, textual information is especially important—it is the most natural, most common, and most expressive form of information.

Text representation plays a critical role in application tasks like classification or information retrieval since the quality of the underlying feature space directly impacts each task's performance. Because of this importance, many different approaches have been developed for generating text representations. By far, the most common way to generate features is to segment text into words and record their n-grams. While simple term features perform relatively well in topic-based tasks, not all downstream applications are of a topical nature and can be captured by words alone. For example, determining the native language of an English essay writer will depend on more than just word choice. Competing methods to topic-based representations (such as neural networks) are often not interpretable or rely on massive amounts of training data. This thesis proposes three novel contributions to generate and analyze a large space of *non-topical* features.

First, structural parse tree features are solely based on structural properties of a parse tree by ignoring all of the syntactic categories in the tree. An important advantage of these "skeletons" over regular syntactic features is that they can capture global tree structures without causing problems of data sparseness or overfitting.

Second, SYNTACTICDIFF explicitly captures differences in a text document with respect to a reference corpus, creating features that are easily explained as weighted word edit differences. These edit features are especially useful since they are derived from information not present in the current document, capturing a type of comparative feature.

Third, Cross-Context Lexical Analysis is a general framework for analyzing similarities and differences in both term meaning and representation with respect to different, potentially overlapping partitions of a text collection. The representations analyzed by CCLA are not limited to topic-based features.

To my parents

ACKNOWLEDGMENTS

I am most grateful to my advisor, Professor ChengXiang Zhai. He is an excellent mentor, and without his help and supportive guidance, this thesis would not be what it is today. It is with his generous time and wise insight that I was able to complete the work that appears in this thesis. I'm particularly appreciative of Cheng's consistent positive manner and his ability to see the "big picture" and potential impact of all ideas.

I would also like to thank my committee members: Professor Julia Hockenmaier, Professor Dan Roth, Professor Steve LaValle, and Professor Qiaozhu Mei. I had the pleasure of taking four semesters of Julia's classes in a row—most of my interest in natural language processing and machine learning stems from them. Dan also sparked my early interest in machine learning when I took classes from him at the Data Sciences Summer Institute, and he greatly helped in my graduate school application to UIUC. I was assigned Steve as my undergraduate mentor during sophomore year and have received guidance from him over a period of almost seven years. I'm thankful for his unique perspective while serving on my committee. I've significantly benefited from Qiaozhu's accessibility during the last year of my thesis; he has offered much practical advice for both work and school. Most of my research is inspired by his work in contextual text mining.

Members of UIUC and industry have also positively impacted not only my research, but also my academic life. A very sincere thanks to Dr. Cinda Heeren, Professor Hari Sundaram, Dr. Jeff Ludwig, Tod Courtney, and Dr. Nate Nichols.

My friends in graduate school have also generously shared motivation, encouragement, and fruitful discussion. Special thanks to Jason Cho, Chase Geigle, Urvashi Khandelwal, and Chase Duncan, among many others. Interacting with you was the key to success in grad school.

Finally, I'd like to extend my gratitude towards my family, especially my wife, Kai Nemoto. This thesis is dedicated to my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

MOTIVATION AND IMPACT

In this chapter, we motivate and declare the thesis topic. We start with a broad, general view of text data mining before focusing on specific contributions and main research questions. The organization of the thesis then follows.

1.1 Text Data

A massive amount of online information is natural language text: newspapers, blog articles, forum posts and comments, tweets, scientific literature, government documents, and more. While in general, all kinds of online information is useful, textual information plays an especially important role—it is the most natural, most common, and most expressive form of information for the following reasons (Zhai and Massung, 2016).

1. Text—transcribed human language—is the **most natural** way of encoding knowledge. As a result, most human insight is encoded in the form of text data. For example, scientific knowledge almost exclusively exists in scientific literature, while technical manuals contain detailed explanations of how to operate devices.

2. Text is by far the **most common** type of information encountered by people. Indeed, most of the information a person produces and consumes daily is in text form.

3. Text is the **most expressive** form of information in the sense that it can be used to describe other media such as video or images. Indeed, image search engines such as those supported by Google and Bing often rely on matching companion text of images to retrieve "matching" images to a user's keyword query.

In contrast to structured data, which conform to well-defined schemas and are thus relatively easy for computers to handle, text has less explicit structure, so the development of intelligent software tools to understand content encoded in text is a necessity. The explosive growth of online text information has

created a strong demand for the following two related services to help people manage and exploit big text data.

First, is text retrieval. The rapid growth of online text information means that no one can possibly digest all the new information created on a daily basis. Thus there is an urgent need for developing intelligent text retrieval systems to help people get access to relevant information quickly and accurately, leading to the recent growth of the Web search industry. Web search engines like Google and Bing are now an essential part of our daily life, serving *billions* of queries each day[1].

Second, is text mining. Due to the fact that text data are produced by humans for communication purposes, they are generally rich in semantic content and often contain valuable knowledge, information, opinions, and preferences. As such, they offer great opportunity for discovering various kinds of knowledge useful for many applications. For example, it is now the norm for people to tap into opinionated text data such as product reviews, forum discussions, and social media text to obtain opinions about topics interesting to them and optimize various decision-making tasks. Once again, due to the overwhelming amount of information, people need intelligent software tools to help discover relevant knowledge.

1.2 Text Representation

Text representation plays a critical role in these downstream tasks since the quality of the underlying feature space directly impacts each task's performance. Because of this importance, many different approaches have been developed for generating text representations.

We can represent text by a set of extracted features, often stored in a feature vector. The simplest features can be the words in the text—e.g., delimited by whitespace. Text representation is a fundamental issue in all text retrieval and text mining applications such as Web search, text clustering, and text categorization. Text representation is also critical for generating useful features to be used in many machine learning algorithms to support natural language processing applications.

In other words, virtually all text mining applications operate on feature vectors of text. The quality and generality of these feature vectors directly impacts the performance of downstream applications. Even the most sophisticated machine learning algorithm will achieve poor results if the feature vectors it operates on are not a sufficient representation of the knowledge encoded in the original text data.

The issue of text representation is complicated because different tasks tend to require a somewhat different perspective of representation—thus a different feature set. For example, while function words (such as *while*, *throughout*, and *despite*) are generally not useful for topic categorization, they may be useful for an authorship attribution categorization task, which may also benefit from features capturing

[1] `http://www.internetlivestats.com/google-search-statistics/`

sentence length and structure. It is therefore important to develop a rich set of potential features that can represent text from orthogonal perspectives and to understand what kind of features are most effective for which tasks.

By far, the most common way to generate features is to segment text into words and record their *n*-grams; indeed, unigram words are a common baseline for information retrieval and text classification applications. Competing methods will challenge the performance and assumptions made by baselines like *n*-gram words.

Other feature types rely on natural language processing, such as part-of-speech tags or grammatical parse tree features. Using topic models like LDA (Blei et al., 2003) or its many variants can be used to represent documents in a lower-dimensional topical space by looking at word co-occurrence statistics. As we use more sophisticated NLP methods to represent text (such as named entity recognition or semantic role labeling), we obtain more informative and often discriminative features, but at a higher cost of introducing errors into the pipeline. On the other hand, shallow features like *n*-gram words are very robust, but may not be able to capture sufficiently characteristic properties. Thus, developing general, robust, and discriminative features is a difficult and important open research problem.

Recently, representation learning—deep learning—as described by Bengio et al. (2013) has become a powerful way to compose features in new dimensions. Deep learning has increased performance in many areas of computer science, such as speech recognition, vision, and even NLP (Collobert et al., 2011). Representations such as paragraph vectors (Le and Mikolov, 2014) and skip-thought vectors (Kiros et al., 2015) can represent entire documents or phrases in a low-dimensional space. Both are built on top of word embeddings such as WORD2VEC (Mikolov et al., 2013b), though these particular word embeddings are not strictly "deep learning" (Levy and Goldberg, 2014).

While intriguing, these deep document representations suffer from a few issues. First, they require much larger amounts of training data compared to more basic methods. Second, due to their complexity, they require very long training times that are usually measured in weeks. Lastly, neural networks are the ultimate black box of decision making. Today, we do not completely understand how they work or how to interpret their results. Bengio et al. (2013) list ten points that make a representation good; unfortunately, *interpretability* is not one of them. In other words, a useful feature representation should be "mineable".

As educators, how can we teach students to write better essays if our autograder is not interpretable? This doesn't solve the workload scaling problem if the instructor or TA still needs to individually meet with each student to explain the results. As members of industry, how can we explain to our bosses that the AI we use won't take over humanity[2]? Understanding *why* a document was ranked higher than

[2] `http://www.businessinsider.com/musk-on-artificial-intelligence-2014-6`

another or *why* a group of documents share a topic is just as important as the search results or topics themselves. Ribeiro et al. (2016) emphasize the importance of human understanding in text mining applications. As a medical practitioner or government agent fighting terrorism, understanding exactly why some models behave the ways they do is critical to their effectiveness and utility in decision making and causal analysis (e.g. Zhao et al. (2017)).

It is therefore important to develop a rich set of potential features that can represent text from orthogonal perspectives. These different perspectives allow us to understand what kind of features are most effective for which tasks, and enable mining knowledge in the underlying datasets that would otherwise be lost in more complicated and less transparent features.

While there is some considerable work in "explaining" existing features (Baehrens et al., 2010; Bansal et al., 2014; Ribeiro et al., 2016), there is less of a focus on designing interpretable, orthogonal features to *n*-gram words in the first place. Ribeiro et al. (2016) introduce methods to explain predictions of classifiers and regressors as well as algorithms to select representative members of each class. They examine text representation with bag-of-words features. While a classifier obtained 94% accuracy on newsgroup categorization, examining highly-weighted unigram word features showed a serious overfitting problem. These are the types of issues that we wish to find through interpretable and understandable features.

To address the limitations of existing work on text representation, this thesis proposes three novel contributions to generate a large space of *non-topical* features from text data in a general, interpretable, and efficient way.

Structural parse tree features (Massung et al., 2013) are one such contribution. These features are solely based on structural properties of a parse tree by ignoring all of the syntactic categories in the tree. An important advantage of these "skeletons" over regular syntactic features is that they can capture global tree structures without causing problems of data sparseness or overfitting. Because of the focus on pure structures, even relatively large skeletons can be observed multiple times in a set of text articles; in contrast, if we were to attach the syntactic categories, we would end up having far more specialized features that may not be observed multiple times in the corpus. When combined with unigram words, skeleton features increased the classification accuracy on two different tasks. Skeletons also provide visual clues as to why they are important for certain classes.

SYNTACTICDIFF (Massung and Zhai, 2015) explicitly captures differences in a text document with respect to a reference corpus, creating features that are easily explained as weighted word edit differences. These edit features are especially useful since they are derived from information not present in the current document, capturing a type of comparative feature. As with our previous work, we used non-native text mining (Massung and Zhai, 2016) as a source of example downstream applications. SYNTACTICDIFF's edit features improved native language identification and were able to be directly used for grammatical

error correction (without a specific, heavily-tuned pipeline). In an essay-grading setting, we grouped students together by the types of mistakes they made (realized as edit features). For example, one cluster of students made many article errors while another overused certain word types.

Cross-Context Lexical Analysis (Massung et al., 2017) directly addresses the issue that good features should be interpretable. We propose a general framework for analyzing similarities and differences in both term meaning and representation with respect to different, potentially overlapping partitions of a text collection. We apply our framework to three different tasks: semantic change detection (discovering words whose meanings changed over time), comparative lexical analysis over context (finding context-sensitive and context-insensitive terms), and word representation comparison (investigating randomness inherent in word embeddings).

In summary, text representation is critical to all text mining applications to avoid overfitting and enable potential causal analysis. This thesis contributes approaches to generate and analyze non-topical features from text data. Novel features (Massung et al., 2013; Massung and Zhai, 2015) are shown to help improve performance for various tasks while simultaneously enabling interesting feature analysis and mining. A novel framework for feature analysis (Massung et al., 2017) that supports non-topical features helps explore and explain text representations, including some that are inherently difficult to analyze, such as word embeddings.

1.3 Organization of this Thesis

This thesis is organized as follows:

Chapter 2 explains and motivates the importance of text data management and analysis (Zhai and Massung, 2016). We explain how text is transformed from a raw string into meaningful features that can be used in downstream applications.

Chapter 3 examines and surveys a subfield of text mining named non-native text analysis (Massung and Zhai, 2016). We see how text mining and representation play a crucial role and have the ability to affect billions of language learners around the world.

Chapter 4 details structural parse tree features for text representation (Massung et al., 2013). We examine these features from a classification perspective and compare representative features conditioned on class labels for native language identification.

Chapter 5 introduces SYNTACTICDIFF (Massung and Zhai, 2015), an operator-based comparative text mining algorithm. We demonstrate SYNTACTICDIFF's performance in three tasks—grammatical error correction, summarization, and classification.

Chapter 6 defines Cross-Context Lexical Analysis (Massung et al., 2017), a general framework for

analyzing similarities and differences in both term meaning and representation with respect to different, potentially overlapping partitions of a text collection.

Chapter 7 explains how the previous methods are all open source and available online as part of the META toolkit (Massung et al., 2016).

Chapter 8 concludes the thesis with a summary of existing work and remaining open questions.

CHAPTER 2

TEXT REPRESENTATION

This chapter gives a very broad, general introduction and literature survey to text representation techniques. Organization and content is an expanded version of section 3.3 from Zhai and Massung (2016).

2.1 Feature Types

Techniques from natural language processing allow us to design many different types of informative features for text objects. Let's take a look at the example sentence:

> *A dog is chasing a boy on the playground.*

We can represent this sentence in many different ways. First, we can always represent such a sentence as a string of characters—this is true for every language. This is perhaps the most general way of representing text since we can always use this approach. Unfortunately, the downside to this representation is that it can't allow us to perform semantic analysis, which is often needed for many applications of text mining. We're not even recognizing words, which are the basic units of meaning for any language. (Of course, there are some situations where characters are useful, but that is not the general case.)

The next version of text representation is performing word segmentation to obtain a sequence of words. In the example sentence, we get features like *dog* and *chasing*. With this level of representation, we suddenly have much more freedom. By identifying words, we can (for example), easily discover the most frequent words in this document or the whole collection. These words can then be used to form topics. Therefore, representing text data as a sequence of words opens up a lot of interesting analysis possibilities.

This level of representation is slightly less general than a string of characters. In some languages—such as Chinese—it's actually not that easy to identify all the word boundaries since that language's text is a sequence of characters with no spaces in between words. To solve this problem, we have to rely on some special techniques to identify words and perform more advanced segmentation that isn't only based on whitespace (which isn't always 100% accurate). So, the sequence of words representation is

not as robust as the string of characters representation. In English, it's very easy to obtain this level of representation so we can use this all the time.

Below, we additionally show three modifications to the whitespace tokenization:

- **Lowercasing** changes capital letters to lowercase ones in order to reduce the total number of unique words and ensure that words like *Cat* and *cat* are counted as the same.
- **Stemming** reduces each word to its root form. This results in a further reduction of unique words and ensures that words like *runs* and *running* are counted the same as *run*.
- **Stopword removal** gets rid of non-content bearing words, which usually do not provide useful information to the downstream task.

These modifications do not greatly change the token-based representation of text. Instead, they slightly adjust the representation in attempts to circumvent its limitations.

A, dog, is, chasing, a, boy, on, the, playground, .	(tokenized)
a, dog, is, chasing, a, boy, on, the, playground, .	(+lowercased)
a, dog, is, chase, a, boy, on, the, playground, .	(+stemmed)
dog, chase, boy, playground	(+no stopwords)

If we go further into natural language processing techniques, we can add part-of-speech (POS) tags to the words. This allows us to count, for example, the most frequent nouns; or, we could determine what kind of nouns are associated with what kind of verbs. This opens up more interesting opportunities for further analysis.

A_{DT} dog_{NN} is $_{VBZ}$ chasing_{VBG} a_{DT} boy_{NN} on_{IN} the_{DT} playground_{NN} $._{.}$

Note that as we add more advanced features, we don't necessarily replace the original word sequence or prior features. Instead, we add this as an additional way of representing the text data. Representing text as both words and POS tags enriches the representation of text data, enabling a deeper, more principled analysis. We will see this theme continued throughout this thesis.

If we go further, then we can apply a grammatical parser to the sentence, obtaining a syntactic structure and productions displayed in Figure 2.1. Again, this further opens up more interesting analysis of (e.g.) writing style or grammatical error correction depending on how features are extracted from the tree and aggregated across sentences.

Going further still into semantic analysis, we might be able to recognize *dog* as an animal. We also can recognize *boy* as a person, and *playground* as a location and analyze their relations. One deduction could be that the dog was chasing the boy, and the boy is on the playground. This will add more entities and

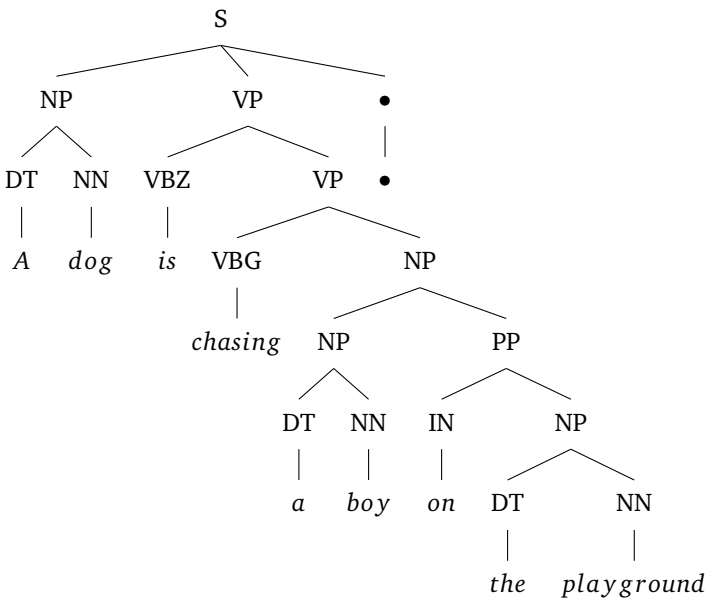

Figure 2.1: An example grammatical parse tree of the sentence *A dog is chasing a boy on the playground*.

relations, through entity-relation recreation. Now, we can count the most frequent person that appears in this whole collection of news articles. Or, whenever this person is mentioned, we discover that we also tend to see mentions of another person or object. These types of repeated pattens can potentially make very good features.

$$\underbrace{\text{A dog}}_{Animal} \rightarrow_{chase} \underbrace{\text{a boy}}_{Person} \rightarrow_{on} \underbrace{\text{the playground}}_{Location}$$

Such a high-level representation is even less robust than the sequence of words or POS tags. It's not always easy to identify all the entities with the right types and we might make mistakes. Relations are even harder to find; again, we might make mistakes. The level of representation is less robust, yet it's very useful. If we move further to a logic representation, then we have predicates and inference rules. With inference rules we can infer interesting derived facts from the text. As one would imagine, we can't do that all the time for all kinds of sentences since it may take significant computation time or a large amount of training data.

Finally, speech acts would add a yet another level of representation of the intent of this sentence. In this example, it might be a request. Knowing that would allow us to analyze even more interesting things about the observer or the author of this sentence. What's the intention of saying that? What scenarios

or what kinds of actions will occur?

Such advanced techniques would require more human effort as well, and they are generally less robust since they attempt to solve a much more difficult problem. If we analyze our text at levels that represent deeper analysis of language, then we have to tolerate potential errors. That also means it's still necessary to combine such deep analysis with shallow analysis based on (e.g.) sequences of words. Despite this, the advanced techniques achieve a representation of text is closer to the knowledge representation in our mind.

Clearly, there is a tradeoff here between doing deeper analysis that might have errors but would give us direct knowledge that can be extracted from text. Doing shadow analysis (which is more robust) wouldn't give us the necessary deeper representation of knowledge.

A further issue is feature sparsity—using individual words to represent meaning results in a very sparse feature space. For example, a typical text corpus may have a vocabulary size of over $100,000$ unique terms. Lowercasing, stemming, and stopword removal were shown as ways to attempt to alleviate this issue, but they will never completely solve the problem. For example, the words "dog" and "canine" will never be the same (or similar) feature in a unigram words representation even though they have an almost identical meaning.

A way to address this sparsity issue is to transform the one-hot $|V|$-sized space into a lower-dimensional dense space. After removing stopwords, our sentence could look like the following:

$$\begin{array}{cccc} \textit{dog} & \textit{chasing} & \textit{boy} & \textit{playground} \\ \begin{pmatrix} 0.0540 \\ 0.0137 \\ 0.0188 \\ \vdots \\ -0.0377 \\ 0.9897 \end{pmatrix} & \begin{pmatrix} -0.1163 \\ 0.0568 \\ 0.1330 \\ \vdots \\ 0.5839 \\ 0.0400 \end{pmatrix} & \begin{pmatrix} 0.1514 \\ -0.0390 \\ 0.0354 \\ \vdots \\ -0.0173 \\ -0.3011 \end{pmatrix} & \begin{pmatrix} -0.0138 \\ 0.0972 \\ 0.0164 \\ \vdots \\ 0.1104 \\ -0.0011 \end{pmatrix} \end{array}$$

Now, (ideally) words such as "dog" and "canine" have higher similarity than "dog" and "banana" by using cosine similarity between their embeddings.

Methods to induce word embeddings have seen some resurgence in popularity recently (Mikolov et al., 2013b; Pennington et al., 2014), but older techniques such as Brown clustering (Brown et al., 1992) have existed for many years. We investigate some different word embeddings methods in section 6.5.

Still, using individual word embeddings to represent an entire document is an issue. Simply averaging each dimension together is a baseline, but combining them in more structured ways is an area of active research (Le and Mikolov, 2014; Kiros et al., 2015; Kenter et al., 2016). Alternatively, jointly

Representation	Generality	Enabled Analysis	Application Example
String	⋆⋆⋆⋆	string and character processing	compression
Words	⋆⋆⋆	word relations, topic and sentiment analysis	thesaurus discovery and opinion mining
+**Syntactic structure**	⋆⋆	syntactic graph analysis	stylistic analysis and structure-based feature extraction
+**Entities/Relations**	⋆	knowledge graph analysis, information network analysis	knowledge/opinion discovery for specific entities
+**Logic predicates**	⋆	integrative analysis of scattered knowledge; logic inference	knowledge assistant for biologists

Table 2.1: Text representation and enabled analysis. More sophisticated representation generally enables more intelligent text analysis applications, but also tends to be less robust. (Reproduced from Zhai and Massung (2016))

learning embeddings along with some neural network task is a possibility, as demonstrated (in one case) by Collobert et al. (2011).

Text data are generated by humans and are meant to be consumed by humans. As a result, in text data analysis and text mining, humans play a very important role. They are always in the loop, meaning that we should optimize for a collaboration between humans and computers. In that sense, it's okay that computers may not be able to have a completely accurate representation of text data. Patterns that are extracted from text data can be interpreted by humans, and then humans can guide the computers to do more accurate analysis by annotating more data, guiding machine learning programs to make them work more effectively.

Different text representation tends to enable different analyses as shown in Table 2.1. In particular, we can gradually add more and more deeper analysis results to represent text data that would open up more interesting representation opportunities and analysis capabilities. The table summarizes what we have just seen; the first column shows the type of text representation while the second visualizes the generality of such a representation. By generality, we mean whether we can do this kind of representation accurately for all the text data (very general; more stars) or only some of them (not very general; fewer stars). The third column shows the enabled analysis techniques and the final column shows some examples of applications that can be achieved with a particular level of representation.

As a sequence of characters, text can only be processed by stream processing algorithms. They are very robust and general. In a compression application, we don't need to know word boundaries (although knowing word boundaries might actually help). Sequences of words—as opposed to characters—afford a very important level of representation; it's quite general and relatively robust, indicating that it supports

many analysis techniques such as word relation analysis, topic analysis, and sentiment analysis. As you may expect, many applications can be enabled by these kinds of analysis. For example, thesaurus discovery has to do with discovering related words, and topic- and opinion-related applications are enabled. People might be interested in knowing major topics covered in the collection of text, where a topic is represented as a distribution over words.

Moving down, we'll see we can gradually add additional representations. By adding syntactic structures, we can enable syntactic graph analysis; we can use graph mining algorithms to analyze these syntactic graphs. For example, stylistic analysis generally requires syntactical structure representation. We can also generate structure-based features that might help us classify the text objects into different categories by looking at their different syntactic structures. If you want to classify articles into different categories corresponding to different authors, then you generally need to look at syntactic structures. When we add entities and relations, then we can enable other techniques such as knowledge graphs or information networks. Using these more advanced feature representations allow applications that deal with entities.

Finally, when we add logical predicates, we can integrate analysis of scattered knowledge. For example, we can add an ontology on top of extracted information from text to make inferences. A good example of an application enabled by this level of representation is a knowledge assistant for biologists. This system is able to manage all the relevant knowledge from literature about a research problem such as understanding gene functions. The computer can make inferences about some of the hypotheses that a biologist might be interested in. For example, it could determine whether a gene has a certain function by reading literature to extract relevant facts. It could use a logic system to track answers to researchers' questions about what genes are related to what functions. In order to support this level of application, we need to go as far as logical representation.

2.2 Using Extracted Features

Consider the following movie review from IMDB[1]:

> *I also am utterly bemused to see so many negative comments on this show. For those who seem to think the show is about pointing out the improved morals of the 21st century, or don't catch the story lines as being evolved enough, or think the characters shallow—I'm afraid you're missing the picture completely.*

Is it positive or negative?

[1] `http://www.imdb.com/title/tt0804503/reviews`

Next, consider the following snippet from an essay (Ishikawa, 2013):

> *I agree this opinion that part-time job is important for college school students, because I studied a lot of thing with my part-time job. At first, the communication skill is necessary to work.*

Was this essay written by a Chinese, Japanese, or English student?

These are examples of categorization problems in text mining. In the first snippet, we want to perform sentiment analysis; that is, we want to determine if the review is positive or negative. This particular review is challenging because it is a positive review about the TV show *Mad Men*, but it is very negative towards the other reviewers. Thus, the sentiment target is critical to capture.

The second snippet—the native language identification task—has its own set of challenges. All the essays are written about similar topics, so "job" and "college" frequently occur given any native language. Perhaps modeling grammatical errors or colloquialisms could better capture information that a classifier can use to separate the classes.

Despite these differences in downstream tasks (and most other tasks), the unigram words representation remains an important baseline to compare against for the same reasons mentioned in chapter 1. Unigram words are effective and interpretable since they can sufficiently capture topical ideas from text (e.g., a document about sports is more likely to contain the word *score* than a document about food). Unigram words are efficient since the representation is achieved by simply splitting on whitespace or following some simple tokenization rules, such as separating *can't* into *can* and *'t*. The generality of words is also clear, since it can be applied to any text document.

The downside here is that due to its great generality, unigram words fail to capture deeper meaning from documents such as the sentiment target. Even more unsatisfactory is the loss of any word order under this model: the sentences *"They have many theoretical ideas"* and *"many They theoretical have ideas"* would be represented in exactly the same way. This is where the need for research into different text representations arises.

Now that we have examined the different types of features, how exactly are they used? Downstream applications make use of document representations (i.e., extracted features) in three main ways: feature vector similarity, feature presence and absence, and feature sequences.

2.2.1 Feature Vector Similarity

Feature vector similarity is crucial in information retrieval and clustering. For example, consider a document d_i from a large collection D. In the vector space model of information retrieval, we assume that relevance can be modeled by similarity. That is, the most similar documents to a given query are the

ones that should be returned by a search engine.

Consider the following:

$$sim(d_i, q) > sim(d_j, q).$$

Here, d_i is found to be more similar (i.e., relevant) to the query q, so it would be rewarded with a higher position on the search engine results page than d_j. Note that both the similarity function and feature representation are user- or algorithm-defined.

A basic similarity algorithm is cosine similarity, which measures the cosine of the angle between two vectors in a high-dimensional feature space,

$$sim_{cosine}(a, b) = \frac{a \cdot b}{||a||\ ||b||} = \frac{\sum_{i=0}^{n} a_i b_i}{\sqrt{\sum_{i=0}^{n} a_i^2}\sqrt{\sum_{i=0}^{n} b_i^2}},$$

but many others exist that are tailored towards information retrieval, such as Okapi BM25 (Robertson et al., 1994).

Clustering also uses similarity measures to assign documents into coherent groups (i.e., clusters) that may be inspected by the user for exploratory analysis or knowledge discovery. Multiple variations of clustering exist. Two simple methods are hierarchical agglomerative clustering and divisive clustering.

Hierarchical agglomerative clustering uses a similarity measure to incrementally group documents together until there is only one large group. The merges that lead up to the single cluster form a dendrogram (cluster tree) that may be traversed or partitioned as desired. Nodes near the root of the tree are more general; as one traverses down the tree, the nodes represent smaller and more specific clusters.

A divisive clustering called K-means uses the expectation maximization framework to optimally partition groups of elements into K clusters, each represented with a cluster centroid. Here, we minimize the distance (i.e., maximize the similarity) between each element and its centroid.

In both clustering cases, the choice of features determines the meaning of the uncovered clusters. For example, using unigram words may find topical clusters while grammatical features may find clusters that are more stylistically-oriented. For a more in-depth discussion of feature usage in information retrieval and clustering, please consult Zhai and Massung (2016) or Han et al. (2011).

2.2.2 Feature presence, absence, and co-occurrence

Feature presence, absence, and co-occurrence is crucial in topic modeling and frequent pattern mining.

Topic modeling is a text mining tool that is used to discover latent themes in a collection of documents. A topic is a distribution of words; additionally, each document is imagined to have some mixture of topics. For instance, a news article could be about both finance and entertainment. The generative process to

write one word in a document is as follows: select a topic $z \sim Multinomial(\theta)$; then, select a word $w \sim Multinomial(\phi)$. Typically, θ and ϕ will have Dirichlet priors.

Latent Dirichlet allocation (Blei et al., 2003) is a well-known topic modeling algorithm. It and similar algorithms learn the θ and ϕ parameters through some form of smoothed counting. In this framework, frequently co-occurring words are likely to become part of a topic.

An important aspect of topic modeling is that the basic units of meaning do not necessarily have to be words—any "bag-of-features" may be used. For example, when clustering student essays, it may be beneficial to represent essays as a collection of grammatical errors. Thus, a topic in this corpus is frequently co-occurring errors or idiosyncrasies, and documents are some mixture of common errors.

Frequent pattern mining (or association rule learning) is a knowledge discovery technique for common sets of unordered items. A standard example is the grocery store purchase history over thousands of customers. It could be discovered that beer, nuts, and diapers are often bought together. This fact can be used to strategically place products in the store in close proximity, encouraging their joint purchase.

Agrawal and Srikant (1994) describe a pattern mining algorithm that generates candidate sets of length k given previously-discovered sets of length $k-1$. This refinement is continued until some confidence threshold is broken, and the user is left with the largest frequent patterns that satisfy some probability constraints. In an alternate approach focused on efficiency, Han et al. (2000) recursively prune a prefix tree of connected items to find the frequent patterns.

Again, if these pattern mining algorithms are run over a text collection, the types of features used to represent the contents of each document will directly impact the meaning of the output.

2.2.3 Sequences of features

Sequences of features are also used in pattern mining as well as NLP tools such as parsers.

In contrast to frequent pattern mining, *sequential* pattern mining finds ordered sequences of items. These items may be actions, events, or other time-based occurrences. If we consider a document as a sequence of features over time, we could use sequential pattern mining to find common phrases or even consecutive topic mentions (e.g., a news reporter may frequently use the ordered transition $sports \rightarrow money \rightarrow politics$).

Pei et al. (2001) is one example of a sequential pattern mining algorithm that focuses on efficiently finding long sequences by using prefix-projection to reduce the number of candidates generated. Such an algorithm that focuses on long sequences could be used (e.g.) to mine characters in streams of text. Focusing on characters instead of words is more robust to word misspellings and other grammatical errors Tsur and Rappoport (2007).

Pattern mining itself can even be thought of as feature generation, as frequent patterns can be used to represent documents in a corpus and fed into a classifier.

Discriminative models for part-of-speech tagging like conditional random fields (Lafferty et al., 2001) allow arbitrary features to be encoded into the model. Thus, features for a current token w_i could have specific dependencies on w_{i-1} and w_{i+1}. These features could be vary from "the previous word ended in *ly*" to "the word after this word is a stop word". Clearly, feature choice for these types of models can have a huge impact on performance. Such engineered features are often critical to the performance of NLP tools that perform semantic role labeling or constituency and dependency parsering.

The examples above are just a few of the many potential applications that depend on a meaningful feature representation. Despite this, most cases will fit into one of the three broad categories of feature vector similarity; feature presence, absence, and co-occurrence; and feature sequences. Most of the work presented in this thesis uses applications from the first two categories, but it is natural to extend the feature representations for use in any of the three groups.

CHAPTER 3

CASE STUDY: NON-NATIVE TEXT MINING

In contrast to chapter 2, this chapter explores a particular subfield of text mining that benefits from advances in text representation. The narrative in this chapter is heavily based on Massung and Zhai (2016).

3.1 Introduction

By the year 2020, Robson (2014) estimates that there will be two billion English language learners—already, it is spoken by one quarter of the Earth's population. Some learn in the classroom; some learn online. Some may even learn through their phone or in an online class. Regardless of the medium, computational tools to enhance this educational experience will be valuable. Automatic scoring of essays—not only for grammar, but also fluency—would contribute greatly to second-language learners' understanding. User personalization for online services (including search engines and social networks) would benefit from improved user profiling. More relevant books or news articles could be recommended if the user's background and competency of English were known.

Due to these many motivating examples, research in *non-native text mining* has prospered. This field encompasses any textual task that deals with words written in a language other than the writer's native tongue. We call the native language L1 and the second, learned language L2. Throughout this thesis, we will usually assume that L2 is English, though most (but not all) techniques mentioned here are general and could function with any pair of L1 and L2.

We provide a brief survey of existing work on non-native text mining. First, we discuss non-native grammatical error correction—finding and modifying text to fix errors or to make it sound more fluent. Second, an introduction to native language identification: determining the native language of an author based on text in the second language. Then, we take a brief look at two emerging fields: native fluency scoring and text simplification for non-native speakers. This concludes the literature review component of this thesis.

The current work on non-native text mining generally falls into the four categories mentioned above:

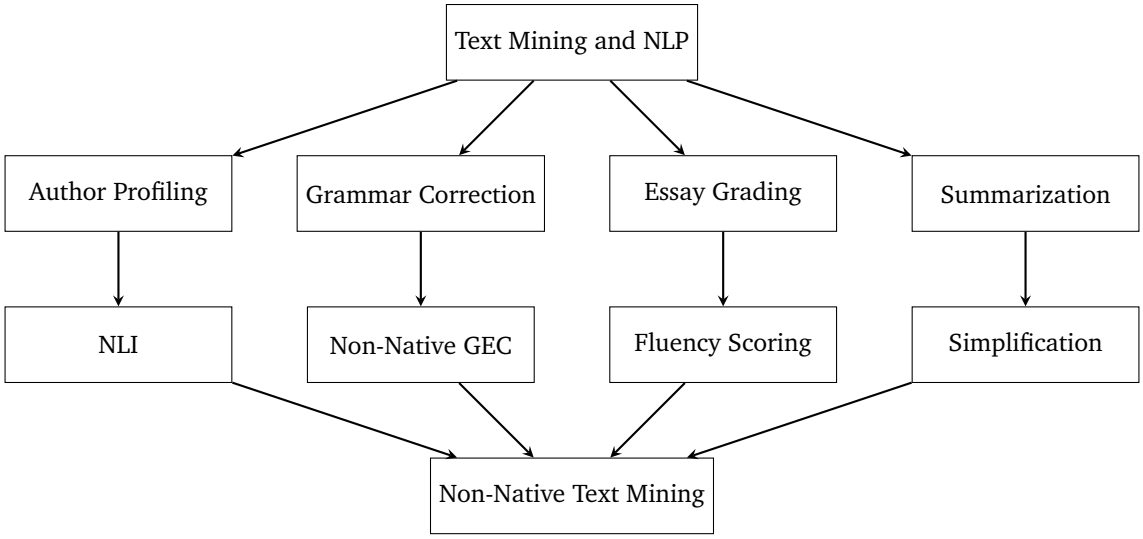

Figure 3.1: Dichotomy of non-native text mining as part of the general text mining and NLP domains.

1. **Native language identification** (NLI): classifying L1 based on text written in L2. Techniques can be categorized into **feature-based** (using a classifier) or **likelihood-based** (using a probabilistic model of language).

2. **Non-native Grammatical Error Correction** (GEC): detecting and correcting grammatical errors in L2 text. Techniques can be categorized into **targeted** (correcting specific errors) or **general** (correcting all errors).

3. **Fluency Scoring**: given L2 text, how close to native does it appear?

4. **Text simplification**: providing a better experience for users interacting with text in their L2. Techniques are much more varied in this field.

These four areas can be regarded as special cases of four more general categories of text mining and NLP as shown in Figure 3.1.

3.2 Native Language Identification

NLI usually relies on classification, but also consists of other components that are able to capture a deeper syntactic meaning (such as dependencies or language modeling). NLI is usually the first step in any second language error correction or author profiling system. Identifying the native language of an anonymous text was first popularized by Koppel et al. (2005). Brooke and Hirst (2012) do an extensive survey of NLI feature efficacy, and develop a robust model that works well when used across corpora.

NLI tasks are most commonly evaluated solely on a small learner corpus usually consisting of student essays. It was previously thought that lexical features would be biased or overfit towards essay topics, but a cross-corpus evaluation showed that this was not the case (Brooke and Hirst, 2012).

For an in-depth discussion of the related task of authorship attribution, we recommend the reader consult Stamatatos (2009). Many techniques common to authorship attribution and author profiling are also relevant to NLI.

Tsur and Rappoport (2007) found that incredibly simple top two hundred frequent bigram character features fed to SVM led to 66% accuracy on the five native languages. They claimed that word choice of non-native speakers is influenced by the phonology of their native language (as evidenced by the effectiveness of the character features). This approach is compared to a unigram words baseline which achieved only 47% accuracy. They finally hypothesized that using a spoken-language corpus would achieve even stronger results favoring character bigrams. Their reasoning was that much less conscious effort is put into speaking than writing. For analyzing transcripts of spoken words, the ICNALE corpus (Ishikawa, 2013) may be applicable.

NLI has also been approached through contrastive analysis (Wong and Dras, 2009): the idea that errors in text are influenced by the native language of the author. They investigated three error types as features: subject-verb disagreement, noun-number disagreement, and determiner misuse. These error types are then used as "stylistic markers" for NLI features with an SVM classifier. To find these errors in text, they used an open source grammar checker[1], as opposed to professionally edited text. Interestingly, ANOVA showed that the features had a measurable effect, but after combining their contrastive features with existing methods, they were not able to significantly increase the classification accuracy from Koppel et al. (2005).

Wong and Dras (2011) follow their work on contrastive analysis, attempting to amend its shortcomings. Instead of error types, they use two different features obtained from grammatical parse trees: horizontal slices (production rules) and parse rerankings. They claim these are the first pure syntactic features used in NLI. For the production rules, they immediately applied information gain dimensionality reduction. The reranking features are those contained in the Charniak parser[2] and Stanford Parser[3] trained on the Wall Street Journal. Unlike the previous two attempts, the authors found MaxEnt to outperform SVM as the classifier. Additionally, five-fold cross validation was performed (as opposed to ten-fold), which means the accuracies cannot be precisely compared with previous work. In any event, they report a final accuracy of 80%, which was the highest reported as of 2012.

Johnson et al. (2006) explore adaptor grammars to generate features. Simply, adaptor grammars are

[1] `http://queequeg.sourceforge.net/index-e.html`
[2] `http://cs.brown.edu/~ec/`
[3] `http://nlp.stanford.edu/software/lex-parser.shtml`

Paper	Method	Accuracy
Tsur and Rappoport (2007)	character *n*-grams*	66%
Wong and Dras (2009)	syntactic errors*	74%
Wong and Dras (2011)	syntactic rules*	80%
Wong et al. (2012)	adaptor grammars*+	76%
Swanson and Charniak (2012)	tree substitution grammars*	78%

Table 3.1: Summary of NLI results listed in Brooke and Hirst for the ICLE corpus; accuracies added to chart. * indicates feature-based methods and + indicates likelihood-based methods.

a non-parametric extension to PCFGs (probabilistic context free grammars). They can learn arbitrary-length word sequences (collocations); for example, *gradient descent* and *cost function* were learned as phrases in a machine learning topic. These adaptor grammars are used in two ways: in the first, collocations are used as features in a MaxEnt classifier. In the second, the grammar is trained on each class (representing native language). At test time, the most probable grammar to have generated the text is selected. For both tasks, the authors use five-fold cross validation on seven native languages. In the feature-based classification, they achieved 76%; in the language model-based classification, they achieved only 50%, a performance similar to the unigram word baseline from Tsur and Rappoport (2007).

Swanson and Charniak (2012) made use of tree substitution grammars (TSGs) Blunsom and Cohn (2010). TSGs are a tree-rewriting formalism that defines operations on partial (parse) tree objects. For example, subtrees may be added or removed from a base tree. Benefits of using this method are priors which prefer smaller production rules and the ability to capture long-range dependencies. Various induction methods are compared to generate features, and five-fold cross validation on seven native languages is performed. All TSG features outperformed the CFG baseline (at 73%). The highest TSG induction method was Bayesian induction at 78%.

In summary, Table 3.1 lists the comparable accuracies from experiments run on the ICLE subset of five European languages. In general though, accuracies between 70% to 80% are standard for a wide variety of techniques and corpora.

3.3 Non-Native Grammatical Error Correction

An excellent overview of grammatical error correction with a focus on non-native learners can be found in Leacock et al. (2014). This short book is a concise collection on the topic and consists of many recent advances since 2010. If the reader wishes to delve into more detail in this subtopic, we suggest referencing their work, whereas this thesis features a broader outline and is thus not able to go into as much depth in one particular area.

Lee and Seneff (2006) train a trigram language model on a lattice of alternatives, where "alternatives" are prepositions, articles, and auxiliaries that may or may not occur between words in the original text. For example, the sentence *I want flight Monday* can be corrected by inserting two tokens as such: *I want a flight on Monday*. Their algorithm first strips all such alternatives from the original sentence. So far, this is not much different from the article and preposition corrections. However, they additionally change each remaining word in the input sentence to be a set of related words to the base form: $want \rightarrow \{want, wants, wanted, wanting\}$. Their language model then outputs the k-best candidates. Next, these candidates are given to a PCFG and reranked. The final output is the top-ranked sentence from the PCFG. Across all experiments, they found that reranking the language model candidates significantly increased the F measure.

West et al. (2011) use bilingual random walks between L1 and L2 word senses. For example, on one side of a bipartite graph are L1 words. There are connections from a word $w \in$ L1 to a word $w' \in$ L2 if a w could be translated into w'. w could be the English word *head*, and be translated into a physical head, head of an organization, or the verb *to head*. This model was used to correct non-native sounding phrases such as *entire stranger* to the more natural *complete stranger*. This bipartite graph was combined with a language model to correct non-native sentences. In these experiments, the native language was Korean. Evaluation was performed with Amazon Mechanical Turk [4] where workers chose between the corrected sentence and the original sentence. Results were not strongly positive, since sometimes the corrected errors changed the meaning of the sentence or made it ungrammatical. In future work the authors suggest using a richer probabilistic model such as a PCFG.

Dahlmeier and Ng (2011b) use the NUCLE corpus to find and correct collocation errors via machine translation. Here, a collocation is a phrase commonly used by native speakers. The authors propose that when a writer mentally translates from L1 to L2, some unnatural phrases result due to word choice. They give an example, "*I like to look movies*" that might be written by a native Chinese speaker since *watch* and *look* are very similar in the L1. It would be possible to correct this to the more grammatical "*I like to look at movies*", but it still does not sound natural. Instead, *look* is replaced by *watch*, resulting in the more fluent collocation *watch movies*. For their experiments, they assume the unnatural collocations have already been identified; this mimics a system where a user may ask for improvement suggestions for a snippet of writing. They train a statistical machine translation model on a parallel Chinese-English corpus to correct collocation errors in the NUCLE corpus. A log-linear model was used to score the candidate phrases which allows additional spelling, homophone, and synonym features to be incorporated. They evaluated their method as a retrieval task, where they returned the top k suggestions to fix each collocation error. Two native-English speakers judged results from five hundred corrections with good rater agreement. Finally,

[4] `https://www.mturk.com/mturk/welcome`

they performed an analysis of errors and found that the main reason top-ranked phrases were not correct was due to out-of-vocabulary words.

Dahlmeier and Ng (2011a) introduce an alternating structure optimization (ASO) approach to GEC. In short, ASO is able to leverage a common structure between multiple related problems. In this case, the related problems are selection (find features from native text) and correction (fix the errors in non-native text). Targets were article and preposition errors, again using the NUCLE corpus. It was shown that ASO significantly outperformed a simple linear classifier as well as two unnamed commercial grammar checkers. Features included part-of-speech tags, hypernyms from WordNet, named entities, and shallow parsing tags.

Gamon (2010) combined a language model and classifier into a "meta-classifier" that detected errors in both article and preposition use. Additionally, they investigated how much more training data is needed for the individual methods to approach the accuracy of the meta-classifier. Features for the classifier were a window of six tokens to the right and left of errors, POS tags, and lexical head features. The classifier was actually split into two steps: first, determine the likely presence of a preposition or article. Then, determine *which* article or preposition should be chosen. The language model is trained on LDC's *Gigaword* corpus and log-likelihood was used (normalized by sentence length). The meta-classifier uses features generated from the two primary models such as ratio of likelihood scores from the language model and classifier decisions. As expected, the meta-classifier outperformed the two simpler models. Prepositions were harder to classify than articles and they required more training data to reach a specific level of accuracy. Future work would be including more primary models to feed features into the meta-classifier.

Madnani et al. (2012) applied "round-trip" machine translation to correct generic errors in L2 text. A round-trip translation used the Google Translate API[5] to translate the candidate text from L2 to eight different languages and back again to L2. Since it is not guaranteed that the translations will preserve the meaning of the sentence, they assert that using a language model to select the most fluent choice is not acceptable. Instead, they combined alignments between the source and each round trip translation to create the final answer. This method had a better likelihood of maintaining the sentence's original meaning. In order to evaluate their system, they had human graders check whether the fixes were fluent as well as retaining the original meaning.

The next two papers we discuss consider alternatives to standard acquisition of training data for GEC. Usually, a dataset annotated corrections is used, but these papers create or acquire their own non-native errors.

Instead of training directly on non-native text, Rozovskaya and Roth (2010) took native text and in-

[5] `http://research.google.com/university/translate/`

tentionally inserted article errors. In order to create a realistic error distribution, they generated the article errors with the same observed distribution as the non-native corpora. Advantages to training data generation in this style are avoiding expensive data annotation, circumventing small corpus sizes, and tailoring the classifier to a particular language. Most of all, their method was shown to be superior to only training a classifier on purely native data. One final note they make is that previous baselines used in the literature are not always appropriate; baselines mentioned before are simply the majority class. While this is acceptable for a selection task, this ignores the actual error rate in the data. Usually, the error rate is much lower (shown to be about 10% in their experiments) than the majority class. Therefore, they argue, a more fair comparison would be the *error reduction*, and this is indeed the measure they use to report their results. They reduced the error detection in native Chinese, Czech, and Russian text by 10%, 5%, and 11% respectively.

Cahill et al. (2013) obtained their grammatical errors from Wikipedia revisions. They filtered the article revisions dataset looking for single-word changes that corresponded to correcting article and preposition errors. Using this method, they obtained over one million corrections. They compared their mined corpus with standard error correction corpora as well as artificially-inserted errors like Rozovskaya and Roth's approach. They found that the larger "somewhat clean" Wikipedia edits were much better in increasing the F_1 of the grammar corrector as opposed to other common datasets, including a smaller "more clean" version of their Wikipedia data. Furthermore, they found that artificial error insertion methods trained on their Wikipedia revisions data increased system accuracy compared to training on smaller corpora, and even generalized across datasets.

In sharp contrast to NLI evaluation, Chodorow et al. (2012) describe many issues regarding evaluation for grammatical error correction, with a focus on non-native sentences. This is a valuable paper for those engaging in any GEC task. Their main issue is the "three-way contingency" between the original sentence, the human correction, and the system output (let alone the evaluation scripts themselves). Mentioned later by Rozovskaya and Roth (2013), this report also emphasizes the significance of the relatively low error rates in non-native sentences—this phenomenon suggests using more interpretable measures such as true and false positives and negatives. Furthermore, how is the severity of an error taken into account? For example, most native English speakers often misuse *who* and *whom*. Their conclusion from these observations is to develop a robust evaluation system based on raw error type counts. This allows bias and error skewness to be perceived by the reader while simultaneously permitting the reader or other evaluator to map the raw error data into another form such as precision, accuracy, prevalence, or bias.

Correcting machine translated text is a related issue, but we do not discuss it here; instead, please see Corston-Oliver et al. (2001) or Gamon et al. (2005b). Table 3.2 compares the different methods discussed in this section.

Paper	Method	Target
Corston-Oliver et al. (2001)	machine translation*	specific injected errors
Lee and Seneff (2006)	LM with PCFG scoring*	articles, prepositions, and word forms
West et al. (2011)	bilingual random walk+	word sense
Dahlmeier and Ng (2011a)	structure optimization*	articles/prepositions
Dahlmeier and Ng (2011b)	machine translation+	collocation errors

Table 3.2: Comparison of GEC strategies. * indicates targeted approaches and + indicates general approaches.

Some grammar correction methods are targeted towards a very specific subset of errors, often categorized by the corpus; others attempt to solve more general errors concerning word sense or collocations. Evaluation for grammar correction is much more varied and unstandardized in comparison to the configurations from NLI as we will see in the next section. Which non-native corpus is used also dictates the types of errors that can be corrected.

3.4 Fluency Scoring

Which of the two following sentences sounds more natural?

1. *"If there are unexpected expenses, material for their lesson, for example, they may not be able to pay money to it only with monthly allowance."*
2. *"If there are unexpected expenses—school materials, for example—they may not be able to afford them with only a monthly allowance."*

Most readers would probably agree that the second sentence sounds much more fluent than the first, even though the first sentence has only very minor grammatical errors. Fluency scoring is thus related to GEC in this way. However, as evidenced above, a lack of grammatical errors does not necessarily mean that a sentence sounds native.

In this section, we will discuss directly approaching fluency scoring in addition to three indirect methods: essay grading, machine translation evaluation, and native vs. non-native sentence classification.

Direct fluency scoring focuses on both the "nativeness", grammaticality, and correct colloquialisms used at an individual sentence level. For example, the following sentence would not be considered totally *fluent* English despite the lack of grammatical errors: *They may not be able to pay money for it.* Of course, grammar does play a very important role in fluency evaluation, but it is not the sole contributor to the final score.

Rabinovich et al. (2016) examine the similarities between native, non-native, and translated texts. They found that non-native and translated texts were much more similar to one another compared to native text. They investigate several features that are particularly indicative of the text source. While they don't use these features directly to measure fluency, it would be straightforward to encode them as features in a fluency scorer. Characteristics they found lexical richness (L2 texts use a smaller vocabulary), collocations (L1 texts use more colloquialisms), cohesive markers (L2 texts overuse sentence transition markers), and personal pronouns (L2 texts are more likely to spell out nouns and proper nouns instead of using pronouns). These would certainly be excellent *explanatory text representations* since all are linguistically motivated and easy for humans to understand.

A main issue with direct fluency scoring is the lack of training or evaluation data. Thus, indirect methods are almost always used instead. Heilman et al. (2014) address the lack of training data by annotating a corpus of TOEFL sentences with a proposed fluency annotation scheme. They train a model to score each sentence on a scale based on the annotations. A main issue with this work is the amount of manual effort required. They produced a corpus of approximately $3,000$ scored sentences and used many hand-crafted features for their predictions. Due to the large number of specialized features operating on a small collection of documents from a very restricted domain, overfitting may be a serious issue.

Yoon and Bhat (2012) use recorded speech by non-native speakers and use part-of-speech n-grams to measure fluency. Of course, there are many differences between spoken and written datasets, but the approach here shows us that relatively simple features can be used quite effectively to score fluency assuming there is enough training data (their dataset had $41,000$ objects). Despite issues that could arise from using imperfect speech recognition and POS-tagging, they achieved high correlation with expert proficiency scores. Finally, they performed a simple feature analysis for which POS-tag sequences are indicative of which class type.

In the next paragraphs, we describe indirect approaches that take advantage of existing methods and corpora to address the fluency scoring task.

Essay scoring is content-based, grammar-based, and could contain a fluency component. Essay scoring could also be a type of discourse analysis that attempts to consider the work as a whole. Often though, simple statistics are taken at the word or sentence level that are assumed to be independent of each other. In some cases, the overall flow of an essay has been investigated (Lynch et al., 2014). Readability scoring Dell'Orletta et al. (2014) is also a related task, which can be used both for fluency scoring and text simplification (see section 3.5). Dikli (2006) gives an informative overview of many different essay scoring systems; for brevity, we only mention a few in this section.

The ETS corpus (Blanchard et al., 2013) is a collection of essays from the Test of English as a Foreign Language (TOEFL), a component of the Educational Testing Service. The TOEFL is an essay-based mea-

sure of English proficiency for students seeking to enroll in college. While the ETS corpus was designed to aid in NLI research, it also has scores in the range $[1, 5]$ for each essay. These scores represent how well the student answered the essay question, and is *not* a measure of fluency. According to the essay scoring rubric[6], even essays with a perfect score may have "occasional language errors" that do not result in "imprecise presentation of content". Similarly, a score of 1 could mean either the response is not relevant to the prompt or the level of English is too low to understand. Thus while the level of English fluency may be correlated with the essay score, the score is strictly a result of how well the prompt was answered.

The e-rater system by Attali and Burstein (2006) was used to grade non-native essays from the ETS dataset. It used features such as "grammar, usage, mechanics, style, organization, length, word length, vocabulary, and correlation between prompt and essay vocabulary", but does not explain in detail how some features are derived. They found that e-rater had the same reliability of two human raters' scores, but the correlation between two human raters was about 0.60, or a weighted kappa rater agreement score of 0.44 (the low end of moderate agreement).

Powers et al. (2001) attempted to undermine the first version of e-rater's scoring abilities by taking advantage of its scoring system. They found it was easy to make e-rater give a higher-than-deserved score, but much harder to make it give a lower-than-deserved score. Even so, these scores are still based on answering the prompt, and not on English fluency.

Machine translation evaluation is another related subfield. Fluency scoring is used to judge the output of machine translation (MT) systems (and vice versa), and there are many metrics that have been designed. Most MT evaluation metrics are designed with a target sentence in mind (since MT is a source to target transformation). BLEU (Papineni et al., 2002) is a common measure that scores each sentence based on n-gram precision and the difference between source and target length. Essay scoring is also used (though to a lesser extent) for MT evaluation; for example, ETS's e-rater system won an MTeval competition Parton et al. (2011).

A simple approach to score fluency without reference translations is to use a language model trained on a large amount of fluent text such as the English Gigaword corpus (CITE). The language model would assign probability to each sentence where more fluent sentences are more probable. On the other hand, Gamon et al. (2005a) describe an MT evaluation system without reference translations that does not rely solely on language modeling.

Classifying native vs. non-native sentences may be used as a component of a fluency scorer or fluency corrector.

Sun et al. (2007) use features called *labeled sequential patterns* (LSPs). Similar to Yoon and Bhat

[6] `https://www.ets.org/Media/Tests/TOEFL/pdf/Writing_Rubrics.pdf`

(2012), their patterns are POS-tags. However, they are discovered via frequent pattern mining (see section 2.2.2). Additionally, they leave in stop words and time words, capturing malformed phrases such as *a NNS*. In addition to these LSPs, they add features about language model perplexity and collocations; however, the LSPs were found to be the most effective features.

Lee et al. (2007) solve the training data sparsity issue by using machine translated text as their L2 text. This agrees with the findings by Rabinovich et al. (2016), where L2 and translated texts are highly similar compared to L1 text. A similar task is detecting whether text is machine-translated Aharoni et al. (2014).

Finally, Horbach et al. (2015) build on POS-tags with stop word features and improve L1 vs. L2 classification results. Particularly useful are similarity scores that can be obtained using their features to show how far apart a given text is from L1. This similarity score could be used in fluency scoring, and they do in fact apply it to coarse-grained adequacy labels (low, medium, or high proficiency). From an explanatory text representation perspective, they can output the most over- and under-used features for each L1. For example, they show native Japanese speakers tend to begin sentences with *first,* while native German speakers tend to begin sentences with *another NN*.

3.5 Text Simplification

Consider the following two sentences:

1. *"The main bar at King's is far older, and is the site of more informal meetings between students. The bar has been traditionally painted a socialist red, including a depiction of a hammer and sickle."*

2. *"King's main bar is older. The bar is traditionally painted a socialist red, including a picture of a hammer and sickle."*

The first sentence is longer and uses a slightly larger vocabulary (*depiction* instead of *picture*). As a non-native speaker of English, it is likely that the second sentence is easier to understand, or would at least take less time to comprehend.

Summarization, simplification, and readability go hand in hand to help a non-native speaker understand text. Unlike NLI, GEC, and even fluency scoring, most algorithms operate solely on well-formed, native L2 passages. Simplification can be seen as an easy-to-understand summary of a more difficult text; simplification essentially "translates" one sentence to another, in efforts to make the result have a better readability. It is a form of monolingual machine translation when using a parallel corpus of advanced and simple language. For a detailed description of general text simplification, we direct the reader to Siddharthan (2014).

Unfortunately, not much work has been done in text simplification *specifically* for non-native speakers. A typical use case is simplifying medical texts so the common reader can make sense of them (e.g. see Abrahamsson et al. (2014)). Other use cases could be helping younger readers or users with learning disabilities.

Wikipedia and Simple Wikipedia[7] are popular parallel corpora. In fact, the first example sentence in this section is from Wikipedia and the second is from Simple Wikipedia. Both Wubben et al. (2012) and Zhu et al. (2010) use them as corpora for sentence simplification via monolingual machine translation. The former uses non-native speakers to judge sentences from their system, but the system itself doesn't take into account the users' native language when forming the simplifications. The latter defines sentence splitting, deletion, reordering, and substitution operations on complex parse trees in order to simplify them into more understandable sentences. They evaluate with standard readability measures as well as perplexity from an English language model.

Lappas and Vlachos (2012) show how to rank documents in a search engine to favor both relevance and readability for non-native speakers. The readability score is determined based on the user's native language, although this is not automatically detected. Each document is assigned a $(relevance, readability)$ pair at query-time, and it can be imagined that documents are plotted in this 2D space. A document is said to dominate another document if it is more understandable and more relevant. In the 2D document space, documents that are not dominated by any other document are on the "skyline" (or perimeter) of the space. These are the documents that are browsed by the user. They evaluated their search engine based on the number of documents a user viewed before satisfaction, and found that taking readability into account decreased the number of documents that needed to be examined.

As the text simplification field continues to evolve, we hope to see more simplification tasks specifically aimed at helping second-language learners. The "teddy bear principle" states that language learners tend to stick with a relatively small set of learned syntactic patterns when speaking or writing in L2. Depending on the L1, a sentence simplification task could translate the complex sentences into a format more comfortable to the user. Petersen and Ostendorf (2007) analyze changes made to professionally abridged versions of newspaper articles to determine common translations. These common modifications could be incorporated in a monolingual translation model.

Another relatively unaddressed question is whether simplification is better than an alternative means to understanding: for example, elaboration. The Master's thesis by Maxwell (2011) considers this question and asserts that elaboration is actually more beneficial based on reading comprehension scores of Korean high school students studying English. She claims that simplification often results in unnatural-sounding phrases that do not resemble authentic L1 text. This is still an open problem that has not been approached

[7] `http://simple.wikipedia.org/wiki/Main_Page`

with computational techniques.

CHAPTER 4

STRUCTURAL PARSE TREE FEATURES

In this chapter, we propose and study novel text representation features created from parse tree structures. Unlike the traditional parse tree features which include all the attached syntactic categories to capture linguistic properties of text, the new features are solely or primarily defined based on the tree structure, and thus better reflect the pure structural properties of parse trees. We hypothesize that these new complex structural features capture an orthogonal perspective of text even compared to advanced syntactic ones. Evaluation based on three different text categorization tasks (i.e., nationality detection, essay scoring, and sentiment analysis) shows that the proposed new tree structure features complement the existing ones to enrich text representation. Experimental results further show that a combination of the proposed new structure features with word n-grams can improve F_1 score and classification accuracy.

4.1 Motivation

The issue of text representation is complicated because different tasks tend to require a somewhat different perspective of representation—thus a different feature set. For example, while functional words are generally not useful for topic categorization, they may be useful for the author attribution categorization task, which may also benefit from features capturing sentence structures. It is therefore important to develop a rich set of potential features that can represent text from different perspectives and to understand what kind of features are most effective for which tasks.

By far, the most common way to generate features is to segment text into words and record their n-grams; indeed, unigrams are quite common for information retrieval and text classification applications.

In addition to content features, functional words and syntactic features have also been considered, notably for tasks such as author attribution or essay scoring. Complementary with content features, syntactic features can better reflect the writing style of an article. For example, simple syntactic features such as n-grams of part-of-speech tags and unigram function words were used for authorship attribution (Stamatatos, 2009; Koppel et al., 2009). To further capture syntactic structures, grammatical productions (rewrite rules) were also discussed as potential features (Baayen et al., 1996), where the authors showed

that rule frequencies were significantly different across classes and used them as features in some simple classification tasks. Later work used syntactic tree features for scoring non-native speech (Chen and Zechner, 2011), authorship attribution (Raghavan et al., 2010; Kim et al., 2011), deception detection (Feng et al., 2012), relation extraction (Jiang and Zhai, 2007), and even tree kernel methods (Agarwal et al., 2011; Zhou et al., 2010).

In this chapter, we propose to investigate a new dimension of text representation based on parse trees with more emphasis on structural representation. Specifically, we propose to define structural features solely based on structural properties of a parse tree by ignoring all of the syntactic categories in the tree. More specifically, we call such new features skeletons to indicate their emphasis on pure structures rather than rewrite rules. A skeleton is defined as any subtree of a parse tree without including any syntactic categories. Compared with syntactic rewrite rules, skeletons can better capture the structural properties of a whole parse tree. Indeed, an important advantage of skeletons over regular syntactic features is that they can capture "global tree structures" without causing problems of data sparseness or overfitting. Because of the focus on pure structures, even relatively large skeletons can be observed multiple times in a reasonably large set of text articles; in contrast, if we are to attach the syntactic categories, we would end up having far more specialized features that may not be observed multiple times in a corpus. We thus hypothesize that skeletons can capture a new additional dimension of text that cannot be easily captured by either content features or regular syntactic features, and thus may serve well as complementary features with the existing ones.

We evaluate the proposed skeleton-based features using three different categorization tasks that likely would benefit from structural representation of text: nationality detection, essay grading, and sentiment analysis. We compare feature combinations of the proposed new features with three common simple features (*n*-grams of words, part-of-speech tags, and function words). We also investigate existing tree features (rewrite rules and syntactic categories), showing that the new skeleton-based features provide orthogonal information compared to the simpler features and validating their usefulness for text representation.

4.2 Related Work

Tree structure has been explored before, though not in a text representation perspective. A treebank described in Black et al. (1996) allows grammatical parse trees to be browsed based on structure alone, but does not provide any sort of classification component. Wang and Neumann (2007) use dependency tree structure in a sentence similarity metric for textual entailment. A sentence similarity measure could possibly be generalized to an entire document, though a purely-structural sentence similarity measure

has not been presented before.

Although both these works consider tree skeletons, they are not used as a feature for text representation, and thus have not been used as features for classification, clustering, or information retrieval.

In previous studies of features for text representation, the authors only examined a small subset of feature competitors: in Raghavan et al. (2010), unigrams, bigrams, and trigrams of words; in Kim et al. (2011), unigram and bigram part-of-speech (POS) tags and bag-of-words function words (FW); in Feng et al. (2012), unigram and bigram words and unigram, bigram, and trigram parts-of-speech. In addition to a limited comparison set, each paper only considered one domain; authorship attribution in the first two, and deception detection in the second. In this paper, we compare these existing features with the proposed new features on three additional tasks: nationality detection, essay scoring, and sentiment analysis.

Chen and Zechner (2011) examine tree features at a much higher level in the form of nonterminals per sentence, e.g. number of noun phrases per sentence and mean number of prepositional phrases per sentence. The work was mainly focused on investigating potential features so no classification tasks were performed.

Kim et al. (2011) mine discriminative frequent PCFG tree patterns for each author. Features used were the rewrite rules and a new pattern, k-embedded-edge (ee) subtrees: subtrees that share a set of k ancestor-descendant subtrees. Therefore, a 0-ee subtree would be one arbitrarily-sized subtree, and a 1-ee subtree would be one subtree and one descendant subtree anywhere in the parse tree. This creates an exponential number of potential patterns, and the authors define algorithms in order to process this large amount of data before pruning the number of ee trees to be used as features. In fact, the algorithms were run on a petascale supercomputer, which justified the implication that their method is quite computationally intensive. Besides the concern of computational complexity, another concern is the high susceptibility of the large number of patterns to overfitting. In contrast, the skeleton features proposed in this paper are efficient to compute and systematically capture the major structures in a parse tree.

Jiang and Zhai (2007) explore feature extraction from sequence (words), syntactic (grammatical parse trees), and dependency (dependency parse trees) subspaces. Features used were n-grams for the word sequences, grammar productions for PCFGs, and dependency paths for the dependency parse trees. They concluded that adding all these features together versus separately only slightly increases performance. We suspect that this is because the structural information encoded in the parse trees is not taken into account.

Grammatical parse tree features have also been explored in classification tasks as tree kernels in Collins and Duffy (2002), Kudo and Matsumoto (2004), and Moschitti (2006). Again, none of this previous work

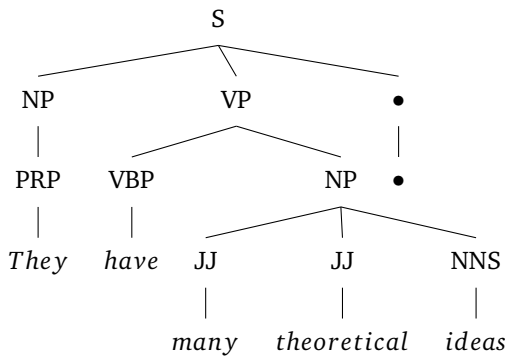

Figure 4.1: An example grammatical parse tree of the sentence *They have many theoretical ideas*.

takes into account the structure of the trees themselves, but rather focuses on the syntactic categories as the main avenue of information. Collins and Duffy (2002) mainly focus on reducing the feature space of "all subtrees" for the perceptron algorithm using rewrite-rule features. Kudo and Matsumoto (2004) explore a boosting method over "subtree stumps" (common subtree sequences). Finally, Moschitti (2006) provide yet another tree kernel method, focusing on efficient algorithms. They use parse tree substructures, a somewhat larger feature space than rewrite rules, but still consider only the node labels in addition to their order.

4.3 Model

A parse tree of the sentence "*They have many theoretical ideas.*" is used for examples and given in Figure 4.1.

The parse tree is rooted with S, denoting $Sentence$; the sentence is composed of a noun phrase (NP) followed by a verb phrase (VP) and period. The leaves of the tree are the words in the sentence, and the preterminals (the direct parents of the leaves) are part-of-speech tags.

It's worth noting that feature extraction from the grammatical structure of a parse tree is separated from the sentence's words themselves. For example, the sentence *"They have many theoretical elbows"*– while nonsensical–will still have the exact same parse tree, since *elbows* and *ideas* are both plural nouns. In both cases, the sentences are grammatically correct.

In this work, we investigate existing parse tree features like syntactic categories and rewrite rules before introducing the novel features *tree skeletons* and *annotated tree skeletons*.

Syntactic category features can be thought of as an extension of POS tags to parse trees. This creates a distribution of non-terminal productions over each class. The trees are simply traversed, tallying the

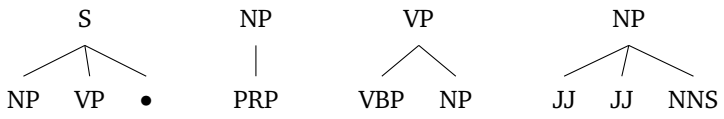

Figure 4.2: Example rewrite rule grammatical parse tree features of the sentence *They have many theoretical ideas*.

labels of internal nodes: S:1, NP:2, VP:1, PRP:1, VBP:1, JJ:2, and NNS:1. The goal of syntactic categories is to observe phrase structure occurrence at a level above POS tags and words. This feature is similar to POS tags in that sense, and it even records them when examining nodes near the leaves of the tree. We hope this feature does at least as well as unigram POS tags, since they are a strict subset of this feature.

Rewrite rules aggregate subtrees from each sentence's parse and was one of the tree features in Kim et al. (2011) and others. The subtrees in Figure 4.2 would be recorded from the example sentence *"They have many theoretical ideas."*:

This process is repeated for all sentences in all texts belonging to a given class, so each class has a distribution of these subtrees. It can be thought of as a "bag-of-trees" method. This feature is desirable, as particular parse trees could be common for any particular category. For example, in age detection, more complicated tree structures could be scarce for younger writers. Similarly, authors whose native language is not English may only select sentence structures from a relatively small learned collection, or repeat similar practiced patterns.

Skeleton features are extracted from a novel procedure that recursively descends into subtrees, recording the internal structure with disregard to internal node labels. This attempts to capture the flow or phrasal structure of sentences while being agnostic to actual labels. The simple COUNTSKELETONS function is described below. SKELETON returns the skeletal structure of the tree rooted at the parameter.

Algorithm 1 Counting different skeletons in a parse tree

```
procedure CountSkeletons(T)
    token ← Skeleton(T)
    incrementCount(token)
    for each subtree t ∈ T do
        CountSkeletons(t)
    end for
end procedure
```

Frequency counts are kept for each tree skeleton in each sentence in the entire class dataset as indicated by the function INCREMENTCOUNT. The skeleton structure representations can be recorded as sets of parenthesis: `((())(()(()))())`. For example, Figure 4.3 displays the skeletons generated from the sentence *"They have many theoretical ideas"*.

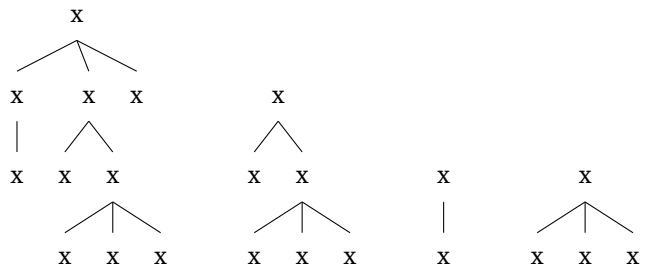

Figure 4.3: Example skeleton grammatical parse tree features of the sentence *They have many theoretical ideas*.

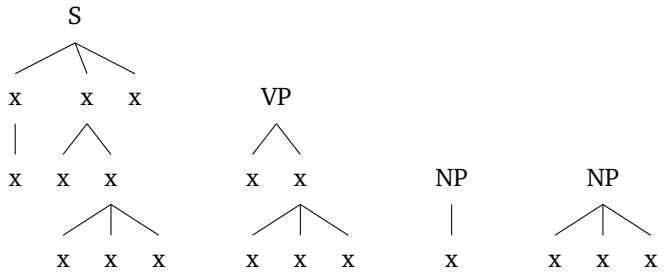

Figure 4.4: Example annotated skeleton grammatical parse tree features of the sentence *They have many theoretical ideas*.

Annotated skeleton features are a compromise between rewrite rules and raw skeletons. They hope to form a middle ground between the specificity of rewrite rules and the generality of skeletons. The algorithm to generate these features is almost exactly the same as skeleton's, except the topmost internal node's label is retained (the annotation). Again, the pseudocode for obtaining annotated skeletons is given below. The function CATEGORY returns the syntactic category of its parameter, e.g. VP or CC.

Algorithm 2 Counting annotated skeletons in a parse tree

```
procedure COUNTANNOTATEDSKELETONS(T)
    token ← CATEGORY(T) + SKELETON(T)
    INCREMENTCOUNT(token)
    for each subtree t ∈ T do
        COUNTANNOTATEDSKELETONS(t)
    end for
end procedure
```

An annotated skeleton feature could be as follows: `(S(())(()(()))())`. Given the example sentence, each subtree above would be given a frequency count of one as shown in Figure 4.4.

A key difference between the skeleton-based features and the existing rewriting rules features is that

the skeleton-based features emphasize pure structural properties by intentionally ignoring the syntactic labels. Thus, they can represent text data from an orthogonal perspective to what existing features can capture. Furthermore, an advantage of such features as compared with rewrite rules is that they are generally more frequent, therefore less likely to suffer from data sparseness.

From another perspective, we may also view a skeleton feature as a cluster of syntactically annotated tree structures that share the same underlying structure. An annotated skeleton is simply a more restricted cluster with the root node fixed to a phrasal category.

As baseline methods, we consider POS tags and function words.

Part-of-speech tags are a common grammatical feature. Their small, finite number lends them to be simple features for a classifier. When expanding to n-grams of part-of-speech tags, their small number also ensures that there are still a relatively low number of features generated (opposed, mainly, to words).

POS tags are perhaps the syntactic analog of basic words, in that they are simple and robust. They capture grammar usage at its most basic level. High accuracy POS taggers ($\geq$ 97%) ensure cleanly processed data.

Function words are a well-performing feature for authorship attribution as noted by Stamatatos (2009) and Koppel et al. (2009). They attempt to capture nuances in text that remain largely an unconscious byproduct of individual authors. In our experiments, we see if our 320 function words also distinguish between nationality, essay grades, or positive or negative sentiment.

Since the n-gram feature generation tools in our toolkit already existed for POS tags and words, we ran the function words collected from the text through this part of the system as well, mainly out of curiosity if bigram function words or higher turned out to be useful.

4.4 Experiments

We evaluate the proposed features using three different text categorization tasks that likely benefit from using structural features.

The CEEAUS (Ishikawa, 2009) dataset consists of 1008 essays written in English by native Chinese, Japanese, and English students. Essays were classified by their writers' native language. In attempts to keep content uniform, each essay is a response to one of two writing prompts: 1) *It is important for college students to have a part-time job* or 2) *Smoking should be completely banned at all restaurants in the country*. Categorizing text based on assumed nationality would be a useful way to rate one's mastery of a second language. It would also aid in authorship profiling when combined with other methods trained on age and gender.

The Essay (Foundation, 2012) dataset is 10,686 scored student essays on a range of 0 to 12. Essays

Nationality Detection (ICNALE)

Method	n_{best}	F_1	A
Word	1	.827	.916
POS-tags	2	.810	.902
Function words	2	.728	.876
Syntactic categories		.703	.844
Rewrite rules		.778	.844
Skeletons		.510	.806
Annotated Skeletons		.721	.870
ASkel + Word	**1**	**.885**	**.942**
SC + Word	1	.867	.936
RR + Word	1	.854	.924
ASkel + POS	2	.811	.900
RR + POS	2	.810	.896
Skel + Word	1	.809	.912

Essay Scoring (Kaggle)

Method	n_{best}	κ
Word	**2**	**.889**
POS-tags	2	.765
Function words	1	.845
Syntactic categories		.431
Rewrite rules		.702
Skeletons		.356
Annotated Skeletons		.658
SC + Word	2	.834
ASkel + FW	1	.822
RR + Word	2	.807
ASkel + Word	2	.791
RR + FW	1	.786
ASkel + POS	2	.782

Sentiment Analysis (IMDB)

Method	n_{best}	F_1	A
Word	1	.820	.820
POS-tags	3	.662	.662
Function words	1	.687	.687
Syntactic Categories		.555	.568
Rewrite rules		.650	.650
Skeletons		.556	.557
Annotated Skeletons		.654	.654
Skel + Word	**1**	**.828**	**.828**
RR + Word	1	.824	.824
ASkel + Word	1	.824	.824
SC + Word	1	.822	.822
ASkel + FW	1	.704	.704
RR + FW	1	.686	.686

Table 4.1: Comparison of single and combined features across corpora. SC, RR, Skel, and ASkel refer to Syntactic Category, Rewrite Rules, Skeleton, and Annotated Skeleton. Combination methods and all parameters were chosen via tuning on development set.

were relatively short, all between 150 and 550 words. These essays were originally used as data for a contest in essay scoring. Scores for the essays are an average of three human graders' scores in an attempt to portray the most accurate human judgement.

The IMDB dataset (Maas et al., 2011a) consists of 50,000 movie reviews from the International Movie Database, classified as either positive or negative. All movie reviews are scored out of 10, but only clearly negative (score ≤ 4) or clearly positive (score ≥ 7) are included in the dataset for data polarization.

To assess classification accuracy for nationality detection and sentiment analysis, we employ the commonly used information retrieval measurements F_1 score and accuracy (Caruana and Niculescu-Mizil, 2004).

The essay dataset is evaluated differently. As in the original contest, performance is calculated with

the quadratic weighted κ metric, described in Cohen (1968). In short, the κ metric measures agreement between two raters using a fixed scale, where usually $\kappa \in [0.0, 1.0]$, with 0.0 indicating random agreement and 1.0 indicating exact agreement. For very poor features, it is possible that the score drops below 0.0. The contest's own evaluation script was run on our output.

Experiment Design

The main questions we strive to answer are Q_1) Are the new features orthogonal to the existing ones? and Q_2) Can we combine the new features with old ones to improve accuracy?

In order to conduct fair experiments, we created a modular testing framework that easily allows us to exchange features and datasets. Source texts were preprocessed with the Stanford parser (Klein and Manning, 2003) and part-of-speech tagger (Toutanova et al., 2003), then features were generated based on the preprocessed data. The feature files are then passed to liblinear (Fan et al., 2008), where it learns a classifier and performs five-fold cross-validation to evaluate the results. We used the parameter `-s 1` for all runs, referring to *L2-regularized L2-loss support vector classification (dual)*. This configuration has $C = 1, B = -1, \epsilon = 0.1$.

In reference to authorship attribution, Stamatatos (2009) notes that the "SVM model is able to avoid overfitting problems even when several thousands of features are used and is considered one of the best solutions of current technology". Hence we chose to use SVMs as our classification method, though of course any classifier could be used.

For word features, 433 stop words based on the Lemur toolkit's (Strohman et al., 2005) stop word list are removed. Then, the words are stemmed according to the Porter2 stemmer (Porter, 2012).

We compare the three baseline features (words, POS tags, function words) with the tree features (rewrite rules, skeletons, annotated skeletons, and syntactic categories). We partition each corpus into two parts; on the first, we perform parameter selection via five-fold cross-validation to find the best n for words, part-of-speech tags, and function words. Then, we select the best-performing n from this set and run it, the tree features, and tree features + best n-grams on the second part, again with five-fold cross-validation. Software used to run all experiments presented in this paper is open-source and freely available online.[1]

Results

Table 4.1 shows the evaluation results on the three data sets. Each column is split into three parts; the top records performance for the best-performing single methods for all n-grams. The middle section shows

[1] https://meta-toolkit.org

tree-based method results, and the bottom section shows the best-performing combined methods.

On the nationality dataset, we see that of the single methods, word unigrams performed the best with an F_1 score of .827, where the best tree feature (RR) had .778. Combining features showed annotated skeletons and unigram words proved most effective ($F_1 = .885$).

The contest that originated the essay dataset ended before this work was begun; the first place finisher ended up with a score of $\kappa = .8141$, but this score is not directly comparable to our results since we believe the contest scored entries based on a withheld testing set. For this task, word features performed the best, with none of the structural features being beneficial. This shows that the effectiveness of features clearly depends on the task. Perhaps the essay scoring is more dependent on content rather than writing style.

Similarly to the nationality detection experiment, word unigrams performed the best in the sentiment analysis task ($F_1 = .820$). For combined features, skeleton and unigram words performed the best with ($F_1 = .828$). Maas et al. (2011a) use this corpus, testing with two folds. They achieved $A = 88.89$. Wang and Manning (2012) also cite using this dataset, with $A = 91.22$ on what we assume to be two folds. We note that our results are significantly less, due to using half the dataset for n-gram parameter selection before running the experiments on the other half.

4.5 Discussion

Table 4.2 shows the relative gain obtained by adding the syntactic category, rewrite rules, skeleton, and annotated skeleton tree features to the original methods.

We find that annotated skeletons provide the best performance boost across all three domains. This confirms our suspicions that structural tree information provides the most useful information. We do not believe n-grams of function words have seriously been considered as a feature, but bigrams of function words worked well when combined with tree features for the nationality and essay data sets.

Mining Tree Features

We use the correlation coefficient as described in Ng et al. (1997) to explore the efficacy of the tree features. Looking at the highest weighted features, we should be able to rationalize their appearance. Given the following metrics for a term t and a category c_i we can define the probabilities:

1. $P(t, c_i)$: presence of t, membership in c_i
2. $P(t, \overline{c_i})$: presence of t, non-membership in c_i

Nationality Detection F_1 Gain

Method	n	Skel	ASkel	RR	SC
Words	1	-.018	.058	.027	.040
POS-tags	2	-.045	.001	.000	-.025
Function words	2	-.074	.071	.067	.024
$\bar{x}$		-.046	**.043**	.031	.013
σ		.028	.037	.034	.033

Essay Scoring κ Gain

Method	n	Skel	ASkel	RR	SC
Words	2	-.140	-.098	-.082	-.055
POS-tags	2	.003	.017	-.004	-.099
Function words	1	-.132	-0.23	-.077	-.145
$\bar{x}$		-.090	-.104	-.054	-.100
σ		.080	.123	.044	.045

Sentiment Analysis F_1 Gain

Method	n	Skel	ASkel	RR	SC
Words	1	.008	.004	.004	.002
POS-tags	3	.015	.023	.019	.002
Function words	1	-.005	.017	-.001	-.030
$\bar{x}$		.006	**.015**	.007	-.090
σ		.010	.010	.010	.018

Table 4.2: Adding tree features to the best-performing single features changes F_1 and κ scores across the three data sets.

3. $P(\overline{t}, c_i)$: absence of t, membership in c_i

4. $P(\overline{t}, \overline{c_i})$ absence of t, non-membership in c_i,

Then, with N total documents, the correlation coefficient (CC) can be written as follows:

$$CC(t, c_i) = \frac{\sqrt{N}[P(t, c_i)P(\overline{t}, \overline{c_i}) - P(t, \overline{c_i})P(\overline{t}, c_i))]}{\sqrt{P(t)P(\overline{t})P(c_i)P(\overline{c_i})}}.$$

The correlation coefficient can be viewed as a one-sided Chi-square metric (Zheng et al., 2004). That is, the features selected by correlation coefficient are most indicative of class membership only (as opposed to membership and non-membership).

Structural tree features allow us automatically mine frequent phrase structures per class by collapsing the surface form into a lower-dimensional structured representation. This dimensionality reduction is critical; note how using unigram words or rewrite rules would only capture subsets of the groups we display in Figure 4.5.

For the first Japanese structure, there were over twenty variations of a writer stating agreement or disagreement with the prompt using the exact same phrase structure (including the incorrect usage of *to*

c_i = **English**

Words	**Skel**	**ASkel**	**RR**	**SC**
educ	()	VBG()	NP→(NP)(VP)	VBG
financi	(()()())	DT()	VP→(MD)(ADVP)(VP)	DT
individu	(()()()())	JJ()	NP→(NP)(PP)	:
believ	(()(()()()()))	NNS()	S→(VP)	-RRB-
right	(()())	NP(()()())	PP→(TO)(NP)	-LRB-

c_i = **Chinese**

RR	**SC**
ADVP→(DT)	JJR
ADVP→(DT)(RBR)	RBR
NP→(NP)(,)(SBAR)	$
ADJP→(JJR)	DT
NP→(JJR)	FRAG

Words	**Skel**	**ASkel**
china	(()(()(()(()(()(())))))))	JJR()
knowledg	((((()())(()(()))))()()((()())(()(())))))	RBR()
partjob	((())(()(())((()((())(())))))	ADVP(()()(()((())(()(())))))
chines	(()((()()()()())(()(()())))	SBAR(()((()(()(()(())))))))
hold	(()(()(()()))(()((()()))(()(()(()())))))))	VP(()(()(()(()(()(())))))))

c_i = **Japanese**

Words	**Skel**	**ASkel**	**RR**	**SC**
think	(())	PRP()	NP→(PRP)	PRP
smoke	(()(()))	NP(())	VP→(VBP)(NP)	VBP
seat	((())(()(()())))	VBP()	NP→(NN)	.
money	(()(())(()(()(()()))()))	.()	VP→(VBP)(S)	LS
japan	((())(()(()()))())	LS()	S→(CC)(NP)(VP)(.)	”

Table 4.3: Samples of the highest ranked features for each language as selected by the correlation coefficient metric. Note that the words are stemmed.

and missing articles when using *first*):

{So,And,But,“}{I,we,they}{agree,disagree}{to,with}{first,that,the,this}{idea,statement,opinion,subject}

These phrases were almost always part of the first sentence of the response. Such patterns may reflect the students' L2 learning style to form these types of sentences, and is why these features are valuable in native language identification. Even if such patterns are found by manually inspecting thousands of essays, extracting them would require writing complicated regular expressions. Even using traditional rewrite rules would not be able to capture these nontrivial productions; thus, we see the power of structural parse tree features in this knowledge-discovery task.

The highly-ranked word features are intuitive, especially interesting are “china” and “chines” for c_i = Chinese, and “japan” for c_i = Japanese.

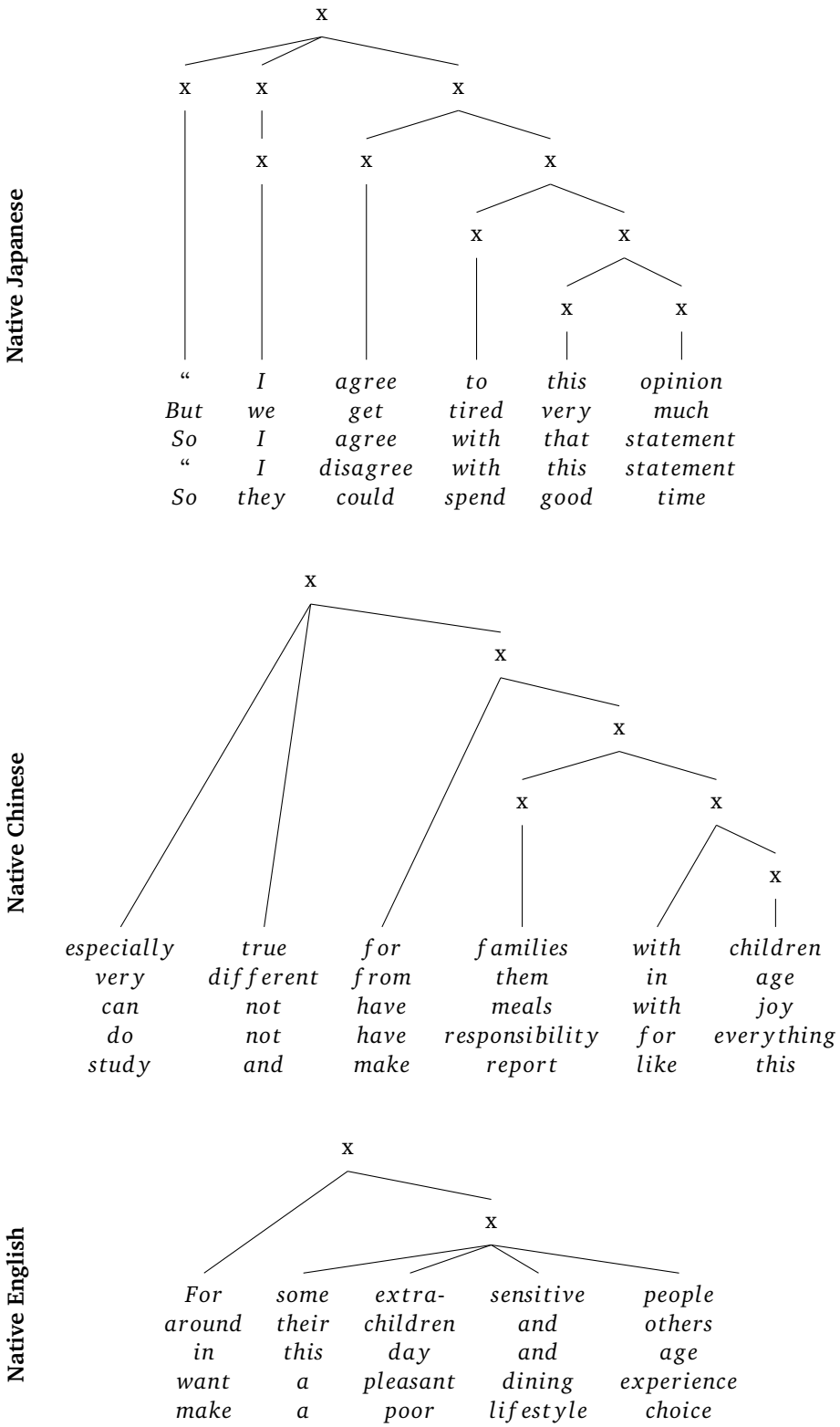

Figure 4.5: Representative phrases for a top feature from each L1.

We have a few observations regarding the syntactic features:

1. Somewhat surprisingly, structural tree features are significantly *shorter* for native speakers.
2. Native speakers use parenthesis (-RRB- and -LRB-) and colons (but not semicolons) much more than non-native speakers do.
3. Non-native speakers more often start phrases with conjunctions.
4. Native speakers use more obscure rewrite rules such as VP→MD, ADVP, VP.

For example, the sentence *"And if there are unexpected expenses, material for their lesson, for example, they may not be able to pay money to it only with monthly allowance*" shows it beginning with a conjunction, and containing a relatively complex (or convoluted) structure.

Another sentence by a native speaker *"Indeed, students who have a part-time job (like I did) quickly change their perspective*" shows a non-standard sentence beginning and the (seemingly) popular parentheses.

These observations are intuitive and lend credibility to the tree features, rationalizing their excellent performance when combined with simple features.

4.6 Contributions and Future Work

We compared combinations of simple n-gram text representation models with new and existing tree features. We showed that the novel structural tree features are most effective and when combined with a simpler lexical model, capturing multiple perspectives of the same text.

Using these new methods, we display performance gains on existing corpora across domains. This demonstrates the generality and usefulness of our features. We showed that the new structural features combine better with simple features than existing tree representations such as rewrite rules.

Additionally, the structural tree features introduced are not restricted to probabilistic context-free grammars as mainly discussed here; they could be applied to other tree structures as well: abstract syntax trees for source code analysis, dependency parses for more linguistic analysis, and even HTML or XML data for Web page or structured document comparisons.

We aimed to answer Q_1) Are the new features orthogonal to the existing ones? and Q_2) Can we combine the new features with old ones to improve accuracy? Based on our experimental results, we can answer *yes* to both. We assert the new features are orthogonal due to lack of syntactic information and positive F_1 score gain after adding them to the lexical features as shown in Table 4.2. We answer Q_2 affirmatively with Table 4.1.

In the future we would like to explore these features in tree structures other than PCFGs, as well in other domains such as clustering and information retrieval. Using structural tree features in a topic modeling context would allow distributions of structures to be obtained for each class more easily than with machine learning algorithms such as SVM. This leads to better interpretability of features, offering clearer explanations of why some classes favor certain structures.

We would also like to compare these new methods with the k-embedded-edge subtrees discussed in Kim et al. (2011), as well as using their proposed feature reduction frequent tree pattern pruning. Additionally, we would be interested in seeing how the features respond to dimensionality reduction techniques, as the number of skeleton and annotated skeleton features is usually quite large.

In work published after this, Nagata and Sakaguchi (2016) examine specific phrase structures from learner English. They create a treebank from non-native corpora that includes interior nodes representing errors such as "word order error" or "word omission error". Their contribution is an improved parser model for learner English and analysis of L2 errors.

CHAPTER 5

SYNTACTICDIFF

In this chapter, we describe SYNTACTICDIFF, a novel, general, and efficient edit-based method for transforming sequences of words given a reference text collection. These transformations can be used directly or can be employed as features to represent text data in a wide variety of text mining applications. As case studies, we apply SYNTACTICDIFF to three quite different tasks, including grammatical error correction, student essay clustering and analysis, and native language identification, showing its benefit in each case. SYNTACTICDIFF is completely general and can thus be potentially applied to any text data in any natural language. It is highly efficient, customizable, and able to capture syntactic differences from a reference text collection at the sentence, document, and subcollection levels. This enables both a rich translation method and feature representation for many text mining tasks that deal with word usage and syntax beyond bag-of-words.

5.1 Motivation

SYNTACTICDIFF was primarily motivated by arguably the most important fundamental question in text data mining: how can we go beyond the bag-of-words representation in a general and robust way? Text representation plays a crucial role in virtually all the text data applications since an inadequate representation always inevitably limits the capacity of a system in performing a mining or analysis task. The most popular text representation used in many applications is the simplest bag-of-words representation, which tends to work reasonably well for many content-processing tasks despite its simplicity. One reason for its popularity is its robustness—it is very general and can be applied to any natural language text. However, such a simple representation is clearly insufficient; for example, it cannot distinguish different orders of words. Improvement over bag-of-words representation has thus been attempted, including n-grams or phrase-based representations, and mixed representations based on part-of-speech tags and words (see section 5.2 for a detailed review of this).

Virtually all the existing work on text representation has assumed that the representation of a text object such as a document would be derived based on *solely the document itself*. Unfortunately, such

an "independent representation" strategy is insufficient for many tasks, particularly those that require discrimination that goes beyond pure content analysis.

For example, to support learning a second language at scale in Massive Open Online Courses (MOOCs), it is often necessary to cluster student essays based on their grammar mistakes to enable "batch grading" of a whole cluster together (Shah et al., 2014). Since all the students may have been asked to write about similar topics, a content-based representation would clearly not work well. To effectively cluster text documents for this application, we would need a representation of each document based on how far it deviates from some reference text data (e.g. writing by native speakers). A comparative analysis of a document with a reference text would be necessary in this case, allowing for the discovery of many subtle differences in the document from comparable native writing. Such a comparative analysis can reveal frequent article errors or incorrect verb form uses, among others. We can use the set of all such mistakes to represent the document in which they occurred, allowing us to cluster essays where similar mistakes are made.

Consider an authorship attribution variant with the goal of identifying the native language of a document's author, a shared task in 2013 (Tetreault et al., 2013). In essence, this is a text categorization problem, so it is common to apply a supervised learning approach. As in the case of clustering, text representation plays a critical role here. Since different authors may have written about the same topic, pure content-based representations again would not work well. Instead, we would need to represent a document based on features that can characterize and distinguish the writing styles. Once again, comparative analysis of the document with a reference corpus of writings by native speakers on similar topics can be very useful for generating more discriminative features to characterize style differences; since writers speaking different native languages tend to have somewhat different writing styles, such features derived from comparative analysis of text are likely much more effective than ordinary content-based features for this categorization task.

In both examples above, we see a clear need for deriving a representation of a text object based on comparative analysis involving another reference text; such a comparative analysis approach to text representation has not been studied in any existing work. In this paper, we conduct the first study of such a new strategy for generating text representation via comparative analysis of text data. Specifically, we propose SYNTACTICDIFF, a novel edit-based method for transforming sequences of words given a reference corpus (model) and use these transformations directly as features or to derive useful features based on them for improved text representation. In addition, the proposed transformation method can be used directly to solve many interesting application problems involving text transformation or comparative analysis of text such as grammatical error correction.

5.1.1 Basic idea of SYNTACTICDIFF

The idea of SYNTACTICDIFF is to define three basic (and therefore general) edit operations: `insert` a word, `remove` a word, and `substitute` one word for another. These edits are used to transform a given sentence. With a source sentence S and a reference text collection R, we can ask the following question: what's the minimum set of edits that we have to apply to S in order to transform it into a sentence in R? This question is interesting because the "minimum set of edits" can be used to measure the deviation of S from sentences in R; what is most interesting is that this "measure" is not a numerical one, but a set of edits that can be features for text representation.

Suppose S is a sentence with potential grammatical errors written by a non-native speaker, and R is a set of sentences written by native speakers on similar topics which includes a very similar sentence to S with no grammatical errors. The minimum set of edits would be very meaningful because they are precisely the *corrections* we must make in order to correct the grammatical errors in S (making it look like it was written by a native). Thus, we can represent the original sentence S with a minimum set of edits, instead of with the words or other content-based features derived from S. Such a transformation-based representation would be much more effective than a content-based representation for generating clusters of sentences that share similar grammatical errors, a task useful for "batch grading" as discussed before.

However, there is one caveat here: what if there is no sentence in R that is very similar to S? We solve this problem by relaxing the requirement of transforming S to a sentence in R and simply requiring the new sentence S^*, resulted from applying a set of edits to S, to "look like" sentences in R. Formally, this can be quantified by estimating an n-gram language model θ based on R, and maximizing the probability of observing S^* from this language model, i.e., seeking S^* that would maximize $P(S^*|\theta)$, or equivalently, minimizing the perplexity of S^* according to θ. This is a very general and robust strategy, as it allows us to compute the minimum set of edits (subject to some constraints on the edits, such as the maximum number of edits allowed) for *any* sentence S with respect to *any* reference text data R. This is similar to likelihood-based methods, but these methods are not created with such rigorously defined operations.

The obtained minimum set of edits can then be used as features to represent text in a context-sensitive way (R as context), which can be used as either an alternative or supplement to the existing content-based representation. By varying the constraints on the edits in interesting ways (e.g. restricting the words to be inserted or deleted to only function words or varying R), we can naturally obtain many interesting variations of text representation that are not possible to generate by any existing methods.

It becomes clear that when restricted to insertion of function words and substitutions involving only lexical transformations, such an edit-based transformation method can be directly useful for grammatical error correction. However, it is important to note that the proposed method can have many other

interesting applications also besides generating interesting features for representing text. For instance, the method can also be used for performing comparative analysis of opposite opinions about an issue in a debate. This can reveal the differences between the opinions since the edits that have to be applied to transform one group of opinions to the other (or vice versa) can potentially reveal the details of their differences. Furthermore, when comparing an article with a reference collection with only deletion edits allowed, we would obtain a set of deletion edits that represent the main topic of the article, since deleting words that are frequent in the article but not frequent in the reference collection is encouraged to make the article conform to the language model induced by the reference collection (those topical words likely have smaller probabilities in the reference collection, thus deleting them in the original article helps increase the likelihood).

5.1.2 Applications of SYNTACTICDIFF

Using the generic framework of SYNTACTICDIFF, we further propose general methods for applying it to three quite different tasks and show that it is beneficial in each case. In the first task, we use weighted word edits with likelihood scoring for grammatical error correction. The method is compared against systems in an grammar correction shared task, and we find that SYNTACTICDIFF edits perform comparably while being much more general than the other methods. In the second task, we create clusters of student essays with similar errors via topic modeling, and find that the interpretability is significantly higher than an n-gram words approach. The third task is native language identification: a classification problem predicting the native language of a student writer based on English essays. We represent documents as vectors of edits, and show that a combination of unigram words and SYNTACTICDIFF edits outperforms each representation individually. In all tasks, we consider SYNTACTICDIFF's efficiency and scalability, showing that is a strong, viable candidate for alternative methods of text representation.

5.2 Related Work

Lee and Seneff (2006) describe a method to correct non-native English sentences. Compared with work in this line, our work, SYNTACTICDIFF, is much more general, since it performs all the basic edit operations (insert, remove, substitute) on real, second language-learner data. SYNTACTICDIFF is also more efficient as it only modifies words in unlikely positions based on a background language model (see section 5.3).

Wong and Dras' contrastive analysis (Wong and Dras, 2009) uses an off-the-shelf grammar checker to generate error-based features. There is no reference corpus or edit-based operations, and it is restricted to a small class of grammatical errors.

SYNTACTICDIFF is related to several other fields, but none of the fields provide full support for all the operations that SYNTACTICDIFF offers. Statistical machine translation (Lopez, 2008) is a related field because one use is to translate non-native language into more fluent language. Unlike our work, comparison of subcorpora is not a natural byproduct of the translation from one sentence to another. Parse tree kernel functions (Moschitti, 2006) can be viewed as assigning a similarity score between a source sentence and target sentence. While this similarity score is quite useful for machine learning problems, it does not provide steps for how to translate the first sentence into the second while maintaining correct syntax. Like tree kernels, DNA sequence alignment from bioinformatics (Li and Homer, 2010) records the similarity between two or more sequences. When applied in an NLP domain, it is to usually solve alignment problems for machine translation (Barzilay and Lee, 2003) or word sense (Barzilay and Lee, 2002). Again, there is no viable method to offer the translation steps while preserving the original structure of the sentence. Additionally, comparing documents using these sequences is not well-defined.

Comparative Text Mining (CTM) (Zhai et al., 2004) uses a mixture model to compare subcorpora. The comparative analysis enabled by this approach is very coarse; in contrast, SYNTACTICDIFF enables very detailed comparative analysis at the level of subtle syntactic and lexical differences.

SYNTACTICDIFF provides a general way to generate new text representations based on a bag of edits, which can be used as alternative or supplementary tokens to feed into *any* topic model as we have explored in the second task. In this sense, SYNTACTICDIFF is *orthogonal to any other text processing techniques*. This opens up many interesting new opportunities for applications and research in text mining.

5.3 Model

SYNTACTICDIFF is a general text analysis framework for transforming (modifying) text with respect to a reference corpus using various edits; the goal is to transform a text object into another so as to better match the reference corpus. Aside from modifying single sentences, it can also be used to make syntactic comparisons between two bodies of text as well as using edits performed on a collection of sentences as features for text representation. We hope to be able to transform, compare, summarize, and induce features from text. The proposed definition of SYNTACTICDIFF will give us the power and flexibility to solve these proposed tasks.

5.3.1 Reference Language Models (LMs)

The reference corpus provides guidance for how we transform a given text object and enables flexible customization of the perspective for defining transformations in SYNTACTICDIFF. The choice of the reference

corpus is thus intentionally application specific.

Given a reference corpus, our goal is to find transformations that can convert any given text object into one that matches the reference corpus as well as possible. Specifically, we would seek transformations to convert the original text object into a new one that would have a higher probability according to the reference language model (LM).

Without loss of generality, we make use of an n-gram LM. An n-gram LM assigns probability to a sequence of m words, where each word is conditioned on the previous $n-1$ words. Thus, for LM θ:

$$P_\theta(w_1, w_2, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-n+1}, \ldots, w_{i-1}).$$

In practice, we reserve probability mass for unseen events by smoothing our LM. A simple form of smoothing used by the SYNTACTICDIFF LM is linear interpolation (Jurafsky and Martin, 2000).

An example of this smoothing for a 3-gram language model is

$$\begin{aligned} P_\theta(w_i | w_{i-2}, w_{i-1}) = \lambda_3 P(w_i | w_{i-2}, w_{i-1}) \\ + \lambda_2 P(w_i | w_{i-1}) \\ + \lambda_1 P(w_i), \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ in order to ensure a valid probability distribution.

Perplexity is a measure for LM evaluation. It can be used to test the likelihood of a sequence given an LM θ.

$$Perp(w_1, w_2, \ldots, w_m) = \left(\prod_{i=1}^{m} \frac{1}{P_\theta(w_i | w_{i-n+1}, \ldots, w_{i-1}} \right)^{\frac{1}{n}}$$

A lower perplexity (or cross-entropy) means that the sequence was more likely to have been generated by θ. We use perplexity per word as a normalized form of scoring for candidate sentences in SYNTACTICDIFF. For a more rigorous and detailed introduction to LMs and their related concepts, please consult Jurafsky and Martin (2000).

5.3.2 Transformation Edits

We define three basic edit operations on sentences:

1. **Insert** the word w after position j in sentence S: $insert(S, j, w)$. The inserted word is drawn from a set of words V^{INS}.

2. **Remove** the word at position j in S: $remove(S, j)$.

Algorithm 3 The SYNTACTICDIFF algorithm

procedure SYNTACTICDIFF(S)
  $candidates \leftarrow \{\}$
  Initialize V^{INS}
  Initialize $V^{SUB}(w) \forall w \in V$
  SYNTACTICDIFF(S, 0)
  return best candidate from $candidates$
end procedure

3. **Substitute** the word at position j in S with w: $substitute(S, j, w)$. The substituted word is drawn from a set of words potentially dependent on w_j: $V^{SUB}(w_j)$.

These three edit functions are used to incrementally transform the original sentence into multiple candidate sentences. The candidate sentences are scored based on perplexity using the reference language model, and the sentence with the lowest perplexity per word becomes the output. Setting V^{INS} to only insert non-content words and setting $V^{SUB}(w)$ to replace words with similar words or inflected forms of the word allows `insert` and `delete` to preserve the original meaning of the sentence, though this is not a requirement. It's possible that the two sets are defined to capture some other grammatical meaning as a particular task demands.

V^{INS} and $V^{SUB}(w)$ may be chosen arbitrarily, thus making size a variable of consideration. In the case where SYNTACTICDIFF is used at large scale on big data, these sets may be reduced to only the most promising elements, in effect reducing the (albeit already competitive) runtime.

For an index j, there are candidates generated from each edit function for a total of $|V^{INS}| + 1 + |V^{SUB}(w_j)|$ edits in addition to the original sentence, which is also regarded as a candidate. Each iteration of SYNTACTICDIFF only performs the edit functions on one index. The index j is chosen by the least likely n-gram from the current sentence $S = w_1, w_2, \ldots, w_m$ (which is most promising for increasing the likelihood and lowering the perplexity). The index of this n-gram is given by

$$j = \arg\max_{i \in [0,m]} \{Perp(w_i, w_{i+1}, \ldots, w_{i+n-1})\}.$$

Next, we need to choose k, the number of iterations to perform. Each iteration operates on all candidate sentences, so for iteration one, only one sentence is operated on. In the second iteration, all new candidates are operated on. Generally, we choose $k \in [1, 5]$ in order to preserve the main content of the original sentence. The full algorithm for SYNTACTICDIFF is given in Alg. 1 and Alg. 2. Initially, we learn an n-gram language model θ from a reference corpus and pick a maximum depth k. Not shown in the pseudocode are checks to ensure edits aren't recomputed for duplicated sentences, since the same candidate sentence may be generated in different branches of the algorithm. This is a simple dynamic

Algorithm 4 The recursive SYNTACTICDIFF algorithm

procedure SYNTACTICDIFF(S, $depth$)
  return if $depth = k$
  $j \leftarrow \arg\max_{i \in [0,m]} \{Perp(w_i, w_{i+1}, \ldots, w_{i+n-1} \in S)\}$
  for $w \in V^{INS}$ **do**
    $S' \leftarrow insert(S, j, w)$
    $candidates.add(S')$
    SYNTACTICDIFF($S', depth + 1$)
  end for
  $S' \leftarrow remove(S, j)$
  $candidates.add(S')$
  SYNTACTICDIFF($S', depth + 1$)
  for $w \in V^{SUB}(w_j)$ **do**
    $S' \leftarrow substitute(S, j, w)$
    $candidates.add(S')$
    SYNTACTICDIFF($S', depth + 1$)
  end for
end procedure

programming optimization.

5.3.3 Weighted Edits

Until now, each candidate sentence is scored equally based on minimizing perplexity per word, regardless of the number or type of edits. This gives the simple scoring function

$$S^* = \underset{S \in candidates}{\arg\min} \{Perp(S)\}.$$

However, we can improve the scoring function to capture some meaning in each edit:

$$S^* = \underset{S \in candidates}{\arg\min} \{\alpha \cdot Perp(S) + (1 - \alpha) \cdot W_S\},$$

where W_S is the edit weight (or edit penalty) of S and $\alpha \in [0, 1]$. α controls the tradeoff between lowering perplexity and lowering penalty; for simplicity, in this first study of SYNTACTICDIFF, we simply set $\alpha = 0.5$ in our experiments, though obviously it is also interesting to further study how to optimize α in the future work. The edit penalty of S can be determined as the average penalty over all edits performed on S. Each penalty edit weight can be on $[0, 1]$.

In this paper, we define four penalties, though the framework is general and any penalty type may be defined using information from the current sentence or reference corpus. We define: an `insert`, `remove`, and `substitute` penalty. We also have a base penalty incurred if *any* edit is performed, penalizing sentences with many edits.

If we set all penalties to zero, we arrive at the original SYNTACTICDIFF formulation; thus, weighted SYNTACTICDIFF is a generalization of the previous description. Furthermore, these penalties can be further refined to vary according to the specific words inserted, deleted, or substituted, and optimized based on specific needs of an application. Since only the scoring function to find S^* changes for weighted edits, the SYNTACTICDIFF algorithm remains unchanged from Algorithm 3 and Algorithm 4.

5.4 Experiments

The proposed SYNTACTICDIFF is useful for a wide range of interesting applications as we will further discuss in Section 5.5. As specific case studies in this section, we apply it to three different and representative text mining tasks related to non-native text analysis in a MOOC or any other online learning scenario. Please note though, that SYNTACTICDIFF could be used in virtually any text mining environment.

First, we show that SYNTACTICDIFF can be used to *search for a transformation* of a sentence with grammatical errors into one with no errors by using native writing as a reference corpus, thus performing *grammatical error correction* as monolingual translation. This application could be a tool that students use to correct or grade their own writing. **Second**, we show that the edits found by SYNTACTICDIFF for each sentence can be used *as new tokens* to replace the original text for topical analysis using topic models. When applied to student essays, this would allow course instructors to find groups of similar essays that share common errors. These clusters can be viewed as a form of *summary of the corpus* and can be used to form teams, pair complementary students, or allow batch grading. **Third**, we show that the edits found by SYNTACTICDIFF can be used *as features* to improve text representation for the *classification task* of native language identification, for which pure content-based features tend not to be very effective. Once a student's native language is known, that information could be used as a fluency score with a confidence level. Additionally, knowing the native language of a student would enable course material to be specifically targeted towards that demographic, or to combat "patriotic grading" (Kulkarni et al., 2013).

Since our goal is to demonstrate the benefit of SYNTACTICDIFF in a variety of different tasks, and due to the space limit, we do not attempt to optimize the performance for any of these tasks and thus do not report detailed results for parameter variations.

All experiments and algorithms are open source and freely available online as part of the toolkit META[1]. The NUCLE corpus[2] (for grammar correction) and the ICNALE corpus[3] (for summarization and classification) are also freely available online. All experiments were run on a laptop with an eight-threaded

[1] `https://meta-toolkit.org`
[2] `http://www.comp.nus.edu.sg/~nlp/corpora.html`
[3] `http://language.sakura.ne.jp/icnale/download.html`

Weight	LM	No-op	P	R	F_1	t
No	No	0.0%	2.96	4.49	3.57	120s
No	Yes	0.8%	3.22	4.47	3.74	**11s**
Yes	No	25.5%	18.78	**19.40**	19.09	123s
Yes	Yes	57.4%	**35.20**	17.55	**23.42**	**11s**

Table 5.1: Grammar correction task: the table shows whether edit weights are used, whether insertions are done based on perplexity, how many final candidate sentences are unchanged (no-ops), precision, recall, F_1 score, and runtime in seconds. This system would place 7^{th} in the CoNLL shared task.

processor and eight gigabytes of memory.

5.4.1 Monolingual Translation

Using the edits directly on each sentence can be seen as a form of monolingual translation. We use the NUCLE corpus (Dahlmeier et al., 2013) to investigate SYNTACTICDIFF's performance in correcting grammatical errors. It is evaluated with precision, recall, and F_1 score using the same framework and testing and training data as the CoNLL-2013 Shared Task in Grammatical Error Correction (Hwee Tou Ng and Siew Mei Wu and Yuanbin Wu and Christian Hadiwinoto and Joel Tetreault, 2013).

Experimental Setup

We used the 1,036 training data sentences to do parameter selection on the four different edit penalties and maximum step size. Since the runtime of SYNTACTICDIFF is quite fast on the NUCLE corpus training data, we easily applied grid search on the weights and k (the maximum number of edits), optimizing the F_1 score. The n-gram value was fixed at $n = 3$, a standard value for sentence fluency scoring purposes. As the reference corpus, we used 50,000 sentences from the Wall Street Journal that are part of the Penn Treebank, since this text is a staple of well-formed English.

The selected edit weights from the training data were 0.0 for substitute and base penalties, 0.07 for insert, and 0.30 for remove. This shows that the default SYNTACTICDIFF needs to remove fewer words to get better performance, while inserting slightly less. The selected value of k was 3. We set V^{INS} to be a short list of function words, since the omission of these is a common error. We used a modified version of the Porter2 stemmer[4] for $V^{SUB}(w)$ that focuses only on reducing plurals and possessives to the same root.

We tested with the designated 345 testing data sentences and used the evaluation scripts from the shared task. Given a candidate sentence S, the predicted corrected form is a new sentence S^* that has the lowest perplexity (see section 5.3.3).

[4] `http://snowball.tartarus.org`

Results

Table 5.1 shows the results of SYNTACTICDIFF used for grammatical error correction, including the precision (P), recall (R), F_1 score, the running time (t), and the percentage of sentences that are unchanged (*No-op*). We also included results without edit positions selected by the language model and results without tuned edit weights. Without edit points selected, edits are performed at every position in the sentence, generating many more candidates. Without edit weights, each edit type is treated equally with no distinction between many or few edits in scoring.

Lack of weighted edits and language model insertion is similar to Lee and Seneff (2006). Of course, the language model is still used to score the candidates in all cases. As seen in Table 5.1, compared with this baseline, the intelligent edit points greatly decrease the run time and the learned edit weights contribute significantly to the performance improvement.

Some sentences in the NUCLE corpus are free of errors, so the correct annotation for these is a no-op. The true no-op rate in the testing data is 36.2%; all other sentences had at least one correction. A system with 100% no-ops received a precision and recall of zero using the CoNLL scoring script. We included the percent of no-ops in the table to compare how zealous each configuration was in suggesting changes. When no edit weights are used, virtually every sentence was modified in some way; consequently, having edit weights ensures that the top-ranked candidate sentence is fluent enough despite having edits.

For a more direct comparison, we can look at the results from the CoNLL shared task where the teams were judged by F_1 score. SYNTACTICDIFF's score of 23.42 would place it in seventh overall, beating out 65% (eleven) of the other teams. Not only does our method place fairly in the shared task standings, but SYNTACTICDIFF is a much more general system than its competitors. The other systems specifically targeted five error types: article/determiner, preposition, noun number, verb form, and subject-verb agreement. The standard system first classified errors into one of the five types. Then, a specific module was run on each error type in order to produce candidates. Finally, the set of candidates were scored, and results from each of the five modules was combined into the final corrected sentence.

SYNTACTICDIFF has no concept of different error types and doesn't rely on classifiers to select particular modules to run. Thus, it is a more general solution than required for the shared task and can be considered *fluency correction*.

5.4.2 Corpus Summarization

Summarizing student essays can give insight into how they are written. Comparable essays will have similar deviances from fluent English. Does a group of students make similar errors? Can we target specific problem areas depending on the group of students we speak to? Or, can we pair students with

complementary strengths and weaknesses?

Topic models such as latent Dirichlet allocation (Blei et al., 2003) are a powerful text analysis tool. After running a topic modeling algorithm, each document in a corpus is assigned a distribution over a fixed number of topics. A topic itself is a distribution over the corpus vocabulary.

We can use the power of topic models to simultaneously cluster and summarize errors in non-native English essays and will show that the bag-of-edits representation enabled by SYNTACTICDIFF is much more useful than the standard bag-of-words representation for this task. For this task, we use the native English essays on a similar topic as the reference corpus; this enables us to use SYNTACTICDIFF to obtain edits more likely related to the usage of English language by the students, which are presumably more useful for this application task than bag-of-words representation.

Experimental Setup

We compare SYNTACTICDIFF edit tokens with unigram and bigram words using the 2,800 ICNALE essays debating public smoking. We hypothesize that the edit tokens will be more interpretable than the competing methods.

Each document is treated as a "bag-of-edits", where SYNTACTICDIFF is run on each sentence in every document. A small feature vector for a document could be

$$\{insert(the) : 3, substitute(a \rightarrow an) : 1, remove(of) : 2\}.$$

We run META's LDA on this feature set, examining the resulting distributions of edits and topics. Hyperparameters were set to 0.1, encouraging sparse distributions.

Since the summarization task is unsupervised, we have no clear objective for parameter selection. Thus, we leave the weights at zero. However, based on the observed output, the user is free to adjust the penalties in order to perturb the results in a direction he or she chooses. Perhaps only substitutions are currently of interest. Due to space constraints, we do not investigate further than all zeroed weights. We set $k = 1$ to get the most likely change to the original sentence, and set $n = 3$. We set V^{INS} to the same function word list as the error correction task and used the full Porter2 stemmer for $V^{SUB}(w)$ since there was no requirement for such precise substitutions. The LDA inference is run with a maximum of one thousand iterations, though all three representations converged before this limit.

Features	$\|V\|$	DL_{avg}	Iteration t	500t
Unigram words	11,580	256	3.2s	26.7m
Bigram words	130,411	255	5.4s	45.0m
SYNTACTICDIFF	2,079	15	0.4s	3.3m

Table 5.2: Summarization task: different tokenization methods for 16 topics on the ICNALE smoking corpus. Displayed are vocabulary size, average document length, LDA inference iteration time, and the time for 500 iterations for comparison.

Results

Table 5.2 compares vocabulary sizes and iteration runtime for the LDA inference. Since the SYNTACTICDIFF edits have much lower dimensionality than the word vectors, inference is significantly faster, even on this relatively small dataset. Table 5.3 shows a sample of topics learned from the ICNALE smoking corpus.

We can see the n-gram representations capture more content-based themes while the edit tokens capture syntactic similarities. For unigram words, topic 1 deals with the physiological concerns of smoking. Topic 12 discusses banning smoking in restaurants, while topic 15 is more nationally-focused. Topic 4 may be of some use, suggesting an overuse of personal pronouns.

Bigram words have almost the same interpretability as unigram words. Topic 4 is similar to topic 12 from the unigram model. Each topic is more of a theme, rather than a collection of grammatical differences. We only see positive essay tokens in each topic, as opposed to lacking (missing) ones.

On the other hand, the SYNTACTICDIFF edits give some insight into the syntactic structure of the student essays. For example, consider these excerpts from three different documents: "*Because it is so bad to mom with baby*", "*In restaurant when people...*", "*...go to restaurant to have meal*". Each student has article use errors which `insert(a)` from topic 4 would fix. The word *a* would never appear in an n-gram topic model because it is absent in each of these documents. Such results can also be used to retrieval sample sentences where the errors occurred.

The same three essays also have an overuse of the word *so*, which `remove(so)` from topic 4 would make more fluent: *so bad, so scared, so dead*. In fact, the first essay contains the phrase *so bad* five times in about fifteen sentences. The third essay contains the sentence "*The smoke make many people feel so bad.*" Aside from the *so* issue as before, there is a subject-verb disagreement between *the smoke* and *make*. While other essays may correctly use the verb *make*, these particular essays use it in an incorrect way such that `sub(make->makes)` is a correction.

Unigram Words

topic 1	topic 4	topic 8	topic 12	topic 15
cancer	i	the	restaurant	i
lung	very	of	banned	japan
smokers	my	tobacco	all	ban
disease	he	cigarettes	country	japanese
heart	was	government	agree	a
nicotine	think	quit	reasons	government
cause	don't	increase	in	just
passive	when	decrease	people	think

Bigram Words

topic 1	topic 4	topic 8	topic 12	topic 15
smoke cigarette	restaurant owners	if you	the media	passive smoker
smoking zone	smoking bans	you are	harmful for	active smoker
can make	bars and	you can	responsibility of	active smokers
sick in	customers would	when you	hotels or	in indonesia
global warming	or non	you smoke	cigarette companies	the active
this policy	ban on	for your	have shown	can disturb
public space	in bars	around you	and teenagers	more dangerous
make many	smoke filled	yourself and	or anything	all restaurant

SYNTACTICDIFF

topic 1	topic 4	topic 8	topic 12	topic 15
insert(the)	insert(a)	remove(you)	remove(so)	remove(area)
remove(opinion)	remove(so)	insert(to)	insert(for)	s(seat→seats)
remove(cigarettes)	s(lung→lungs)	s(reason→reasons)	insert(in)	remove(of)
s(give→giving)	s(make→makes)	s(ban→banning)	remove(not)	s(stop→stopped)
remove(bans)	remove(healthy)	remove(us)	s(have→having)	remove(again)
insert(you)	remove(reasons)	remove(person)	remove(nonsmoker)	insert(i)
remove(totally)	remove(as)	insert(are)	remove(that)	remove(all)
s(cause→causes)	remove(even)	remove(better)	remove(increasing)	insert(it)

Table 5.3: Summarization task: 5 of 16 topics learned from the ICNALE smoking corpus with three tokenization methods. The *n*-gram methods capture writing themes while SYNTACTICDIFF captures similar errors. Note that $s(\cdot)$ refers to the `substitute` function for brevity.

5.4.3 Machine Learning

We use the ICNALE native language identification corpus (Ishikawa, 2013) to test the effectiveness of using the SYNTACTICDIFF edits as features to represent text for classifying English essays based on the native language of the author. This corpus contains 5,600 total essays on two prompts. We hypothesize that the SYNTACTICDIFF features capture the grammatical differences in writing styles of the eleven different native backgrounds.

Experimental Setup

The same bag-of-edits representation as the summarization task is used as input for a classifier to predict the native language of the student essay writer. The Wall Street Journal sentences from the Penn Treebank are used for the reference language model as they were for the monolingual translation task.

Features	$\|V\|$	DL_{avg}	F_1	$Acc.$
Unigram words	9,021	129	80.1	81.8
SYNTACTICDIFF	12,279	56	73.1	75.4
Combined	21,300	185	**84.5***	**85.9***

Table 5.4: Classification task: comparison between the three methods on the ICNALE essays. Displayed are vocabulary size, average document length, F_1 score, and accuracy. *Combined results are significantly higher with $p < 0.001$. Each experiment completed in less than 10 seconds.

	CHN	ENS	HKG	IDN	JPN	KOR	PAK	PHL	SIN	THA	TWN
CHN	**92**	0	0	0	1	1	1	1	0	1	3
ENS	0	**90**	1	0	2	2	0	2	1	1	1
HKG	10	3	**64**	1	2	0	1	2	8	5	4
IDN	2	1	1	**83**	0	1	1	3	1	6	1
JPN	1	1	0	1	**94**	2	0	0	0	0	1
KOR	5	1	1	1	7	**76**	1	1	0	6	1
PAK	1	0	1	0	0	1	**94**	2	0	1	0
PHL	4	1	0	0	0	2	1	**84**	2	5	1
SIN	1	2	0	1	0	2	1	5	**87**	1	0
THA	2	0	0	2	1	3	0	1	0	**90**	1
TWN	12	1	2	0	3	6	2	2	1	5	**66**

Table 5.5: Classification task: confusion matrix of combined features on the ICNALE corpus. Overall accuracy of 85.9%. Percentages have been rounded for readability. Each $(row, column)$ index represents the fraction of times row is labeled as $column$; thus all rows sum to 100%.

As a baseline, we use a standard unigram words feature representation with stemming and stop word removal. Additionally, we combine the unigram words representation with the SYNTACTICDIFF features to see if the performance increases compared to using only one method.

The ICNALE corpus is split in half, based on whether the essay is a smoking essay or a part-time job essay. We use the part-time job subcorpus as a development set to do parameter selection on n and k, for the n-gram language model and maximum number of edits respectively. Once the parameters $(k = 5, n = 5)$ were chosen, we evaluated with five-fold cross validation on the smoking testing set. Each fold of the cross validation is used to do an unpaired t-test for statistical significance. For both development and testing, we use the default SVM classifier that is part of META. The unigram words baseline and feature combination are also part of the same toolkit.

Since adding edit weights will always decrease the score of candidate sentences, we set them all to zero for the classification task. We want the learned SYNTACTICDIFF model to have full control over the generated edits that appear as features. In contrast to the monolingual translation task, we prefer to *minimize* the number of no-ops, since each edit operation is used as a feature; more no-ops means less information is represented. The edit weights are easily set if the user requires, e.g. to ignore a particular operation. Finally, we leave V^{INS} and $V^{SUB}(w)$ the same as the summarization task.

CHN	HKG	ENS	JPN	KOR
remove(people's)	remove(hong)	remove(<s>)	remove(seat)	remove(<s>)
remove(china)	remove(kong)	insert(is)	remove(nonsmoking)	remove(korea)
insert(the)	insert(the)	remove(bad)	remove(tobacco's)	remove(sterility)
remove(harmony)	sub(forced→forcing)	insert(a)	sub(so→be)	insert(or)
sub(people's→people)	remove(don't)	unmodified	remove(can't)	remove(rice)
remove(etc)	remove(carcinogenic)	remove(good)	remove(foods)	remove(habit)
insert(such)	sub(affected→affecting)	insert(such)	remove(opinion)	remove(non)
sub(terrible→terribly)	remove(country)	insert(to)	remove(two)	sub(fair→fairly)

Table 5.6: Classification task: edit features selected via information gain for 5 of the 11 classes in the ICNALE corpus.

Results

Table 5.4 shows a comparison between the three methods: unigram words baseline, SYNTACTICDIFF, and a combination. While unigram words does outperform edit features in F_1 and accuracy, a combination is able to increase both measures at a significance level of $p < 0.001$. This shows that the syntactic edit features capture an orthogonal perspective of the student essays compared to the lexical features as we expected.

Table 5.5 shows a confusion matrix of the eleven classes using the combined features. Each row is a distribution over which class label was chosen for the given row name; the diagonal represents a correct categorization. From this, we see that Japanese and Pakistani students are confidently modeled. Students from Hong Kong and Taiwan and more easily confused with native Chinese speakers, which is logical.

The most informative features for some selected classes are shown in Table 5.6 according to information gain (Zheng et al., 2004). Information gain is a commonly-used feature selection metric in the machine learning and information retrieval communities. It describes the difference in entropy by knowing the presence or absence of a specific term appearing in a class.

Some features are obvious and not as informative to the human reader: Chinese and Korean students overuse *China* and *Korea* compared to the reference language model. Less apparent (yet still useful) edits are the Chinese students' overuse of *etc*, the Hong Kong students' underuse of *the*, the Japanese students' mixup between *so* and *be*, and the Korean students' differentiation between *fair* and *fairly*. We also notice that the native English-speaking students have *unmodified* as a main feature, meaning the perplexity-based candidate scoring preferred their original sentences over edited ones.

There are also a few artifacts of the tokenization method; the sentence marker `<s>` appears as a top feature, implying that English and Korean speakers tend to have shorter sentences, at least compared to the reference model.

5.5 Discussion

This section addresses various questions and hypotheses brought up in our study, particularly the generality of SYNTACTICDIFF and the new application and research directions it can potentially open up.

5.5.1 Generality

As a new way of representing text, SYNTACTICDIFF is very general and robust; just like the bag-of-words representation, the bag-of-edits representation can be applied to *arbitrary* text data to obtain interesting variations of text representation. Note that the bag-of-edits representation is *not* meant to compete with the bag-of-words representation, but rather to supplement it to improve text representation since they capture different perspectives of representation.

The applications of SYNTACTICDIFF are not restricted to improving text representation. SYNTACTICDIFF is a general framework, rather than a particular algorithm. Virtually all the components in SYNTACTICDIFF are configurable; most obviously, edit weight values, n-gram settings, and the reference corpus. Edit weights and n-gram values do not necessarily contribute to any specific syntactic meaning. Rather, these settings are for tuning a model against some objective function, which can vary according to applications (e.g., in the grammatical error correction case, we set edit weights to optimize F_1 score).

The reference language model from the reference corpus plays a more important role in the meaning of each edit. It steers the edit transformations in a particular direction, coaxing each candidate sentence to align with the reference. In our experiments, we considered the reference to be gold standard language, since our tasks dealt with non-native English speakers. Modifying each sentence to minimize its distance with well-formed English makes sense. However, there are many ways to choose and set the reference, enabling the support of other interesting tasks.

For example, suppose we operate on a sentiment analysis dataset. We have a reference model of very positive sentences, and use SYNTACTICDIFF to translate candidate sentences to match the reference. Depending on the sentiment polarity of the candidate sentences, do negative sentences have a different pattern of edits than positive ones? It is in this way that the reference language model choice influences the significance of each edit.

In our experiments, we defined four edit weight penalties. In practice, these could be almost anything the user desires. Returning to the sentiment analysis task, imagine an edit weight penalty that is imposed if the words *no* or *not* are inserted. Or, if a word has a positive sentiment affiliation a penalty is also triggered. Finally, what if at each iteration, a penalty is imposed if the edit operation changes the polarity of the sentence? Some of these suggestions require a classifier in the candidate generation stage; alternatively, sentiment valence scores (Pang and Lee, 2008) could be used as a crude (yet effective)

judgement.

Sentence edit features themselves are also configurable; for instance, we could include the previous word or word index. Then `insert(the)` could be come `insert(the|in)` meaning add *the* after *in* or `insert(4, the)` representing add *the* in the fourth position in the sentence.

Due to space constraints, we could not investigate all possible variations described above, but we envision much future work in this direction.

5.5.2 Applications

The three tasks that we have applied SYNTACTICDIFF to only represent a few of the many possible uses, but even these already have a very broad scope:

Text transformation: In the first task, the edits are used directly to search for an optimal transformation of an original sentence. This represents a general new retrieval model that allows us to use the original sentence as a query to "retrieve" a relevant sentence that best matches the query, where "matching" is based on the edits that we allow. By varying the edits allowed, their weights, and the choice of reference language model, this can potentially support many interesting text transformations that can easily go beyond grammatical error correction (like improvement of coherence, retrieval of opposite opinions, or text summarization).

Comparative text mining: In our second task, we used the edits to represent the original text in an unsupervised learning setting (i.e., topic modeling), which enabled discovery of interesting clusters of related edits. It is very easy to imagine the use of this strategy for many other unsupervised learning methods such as matrix factorization. Also, there are many variants of the basic topic models that can perform more sophisticated topic analysis. All these algorithms can be combined with SYNTACTICDIFF to open up interesting new opportunities for comparative text mining.

Improving text representation for machine learning: In our third task, we used edits to represent text in a supervised learning setting, and showed superior performance of such a representation in comparison to existing text representation methods for the task of native language identification. Supervised learning is widely applied in many text processing tasks. Thus SYNTACTICDIFF can be potentially useful for improving text representation for many of these tasks. Note that we do not have to solely rely on edits for text representation, and can in general combine edit-based representation with content-based representation. This would provide an interesting general and robust way to represent text. Moreover, such an improved representation can easily be exploited in the feedback process of a retrieval task where we face the problem of supervised or semi-supervised learning from a set of feedback documents and the representation of these feedback documents can be improved with SYNTACTICDIFF.

We anticipate many more creative uses of this framework in other text mining tasks to be possible.

5.5.3 Semantic Diff

Using a customized SYNTACTICDIFF for each task allows researchers to gain insight into the differences between subsets of a corpus. How much customization is required to push SYNTACTICDIFF to SEMANTICDIFF?

We have already seen how edit penalty types can be imposed in an ad hoc manner, and how their weights can be chosen intuitively. Although we set $V^{SUB}(w)$ to be a stemmer, it could just as easily be a thesaurus or negator, focused on word sense disambiguation.

We can get even more creative knowing the parts of speech of each word. What if we only insert articles and determiners instead of a list of common function words? We can even design penalty weights for the part of speech. Is it more important to remove a determiner than it is a verb? It depends on the application, and can be learned automatically. Although these many possibilities greatly expand the search space, more advanced candidate selection algorithms such as beam search (Norvig, 1992) can easily be applied.

With a basis for penalty creation, it would be possible to create penalty types on the fly during a training phase. We can break the definition of a penalty into context and an argument. For instance, one context could be surrounding part of speech tags, and the argument is the current word examined in an edit operation. Once SYNTACTICDIFF operates in this format, we can arbitrarily create penalties.

Given all these modifications enabling increased generality, we assert that SEMANTICDIFF is not only attainable, but will form the landscape of edit-based rich text meaning.

5.6 Contributions and Future Work

We presented SYNTACTICDIFF, a novel, efficient, and general framework for many text mining tasks that examines syntactic differences between current text and a reference background collection. These differences are captured in weighted edit operations. These text edits can not only be used to generate an alternative representation of text data that is complementary with the content-based representation, but also support a wide range of interesting novel applications.

We evaluated the generality and effectiveness of SYNTACTICDIFF using three distinct tasks: grammatical error correction, corpus summarization, and classification. In all areas, SYNTACTICDIFF provided concrete advantages, clearly demonstrating its empirical benefit. In the first, we achieved remarkable performance considering our generality compared to other systems. In the second, we summarized grammatical er-

rors better than baseline systems. Lastly, we increased the accuracy of a baseline native language identification system by augmenting with SYNTACTICDIFF edit features. Despite its increased performance, SYNTACTICDIFF comes with no runtime performance penalty, and in some cases is faster than the baseline. While the experiments we have conducted in this paper all involve relatively small data sets, the relatively low computational complexity of SYNTACTICDIFF and its support for flexibility scalability and accuracy tradeoffs make it an appealing novel approach to analysis of big text data.

Our exploration in this paper was only the tip of the iceberg concerning SYNTACTICDIFF's great potential; there are many interesting future directions to further explore, particularly in leveraging such a new representation in many other applications, exploring different configurations for comparative text analysis, and further generalizing the framework to capture more semantic meaning—moving from SYNTACTICDIFF to SEMANTICDIFF.

CHAPTER 6

CROSS-CONTEXT LEXICAL ANALYSIS

We propose a general framework for performing cross-context lexical analysis; that is, analyzing similarities and differences in term meaning and representation with respect to different, potentially overlapping partitions of a text collection.

We apply our framework to three different tasks: semantic change detection (discovering words whose meanings changed over time), comparative lexical analysis over context (finding context-sensitive and context-*in*sensitive terms), and word representation comparison (investigating randomness inherent in word embeddings).

6.1 Motivation

Natural language is almost always used in a particular context (e.g., a particular time, location, or purpose), and thus the interpretation of a sentence, phrase, or word inherently depends on this context. Indeed, the whole subject area of pragmatics studies the ways in which context contributes meaning[1]. In this paper, we are interested in analyzing the variations of term meaning in different–but comparable–contexts and propose a general framework for performing cross-context lexical analysis (CCLA). We use CCLA to generally refer to any analysis of term meaning or term representation in different contexts, especially for understanding the differences and similarities in *multiple* contexts.

Due to the generality of the notion of context, CCLA can be useful in many ways. For example, when context is defined as the time period a piece of text is written, CCLA allows us to compare the meaning of a word in different periods and reveal how a word may have evolved over time (Hamilton et al., 2016). If context is defined as location, it would allow us to study variations in the meaning of a word over different locations, potentially revealing influences of some locations on others (Kulkarni et al., 2016).

In general, we can use *any* associated attribute values of text data—including metadata—as context to form a partition. For example, the institution of a research article's author can be used as a "context variable" to partition the articles based on institutions or regions in the world of their authors.

[1] `https://en.wikipedia.org/wiki/Pragmatics`

Any meaningful partitioning of text data may also be regarded as implicitly defining a context value for each partition; sentiment analysis may allow us to define a sentiment context so positive and negative sentences would be regarded as belonging to different categories.

We can characterize any term in a specific context by its similarity to other terms in corresponding contexts. The similarity can be computed in many ways, including (e.g.) with word embeddings. This gives us a context-specific "term similarity profile" for every term. These profiles for the same term computed from different contexts can be compared to analyze the variations of term meaning across contexts.

Traditionally, such cross-contextual analysis has been done on a "topic-level" basis (Zhai et al., 2004; Mei and Zhai, 2006). However, this is limiting because only word co-occurrence data can be used to estimate the model. Thus, including distributional similarity metrics (or any other representation) is not built-in, and it is not obvious how to include it in a probabilistic model in an easily-interchangeable way. Lastly, relying solely on word co-occurrence statistics (which are often unigrams) misses opportunities to examine context windows of adjacent terms, which could be useful for capturing word sense or ambiguity.

CCLA can be used to perform analysis in three distinct ways:

1. a **term focused** approach, where the emphasis is placed on mining the terms themselves with respect to the collection of contexts. For example, we could detect words whose meanings have shifted over time (which we explore in section 6.3), or compare dialects of the same language across different regions;

2. a **score focused** approach, where the emphasis is placed on defining a scoring function over terms that can detect context-sensitive (representative) or context-insensitive (shared) terms (which we explore in section 6.4). This can be useful as a component in downstream tasks such as feature selection, transfer learning, and information retrieval; and

3. an **annotation focused** approach, where the emphasis is placed on understanding how the annotations for words change as a function of the context used to derive the annotation. We explore this in section 6.5, where we analyze the stability of two well-known word embedding methods.

These focuses often intermix and overlap.

This paper is organized in the following manner. Section 6.2 formalizes cross-context lexical analysis. Sections 6.3-6.5 investigate concrete applications of CCLA and illustrate each of the three focuses described above. Section 6.7 shows related work and section 6.8 concludes the paper.

All source code from this work is made publicly available online (Massung et al., 2016). All datasets used in our experiments are also freely and publicly available.

6.2 Model

We now formally define the framework for cross-context lexical analysis. Critical to CCLA is the idea of a context view. We define a context view as a tuple $C_i = (V_i, f_i)$, where V_i is a set of unique terms $w \in V_i$ and $f_i : V_i \rightarrow \mathbb{A}$ is an annotation function that maps words from the vocabulary set V_i to some shared analysis space $\mathbb{A}$. Potential annotations could be term probabilities ($\mathbb{A} = [0, 1]$) or word vectors ($\mathbb{A} = \mathbb{R}^d$), depending on the eventual goal. Different contexts C_i and C_j may share word tokens, but each word's annotation is specific to its context. That is, the term $w = amazing$ may occur in both V_i and V_j, but $f_i(w)$ is not necessarily equal to $f_j(w)$. This allows us to compare the usage of the token $amazing$ respective to each context. We refer to the set of all contexts as **C**.

The vocabularies for each context come from a backing set of text documents D. This may be a corpus in the conventional notion—like the IMDB movie reviews (Maas et al., 2011b)—or it may be a collection of such corpora. Due to this flexible nature of context views, it is not a requirement that all contexts partition D; contexts may even overlap. Take the sentiment analysis dataset collection as an example: imagine that D contains documents from both IMDB and Yelp[2]. If we set $\mathbb{A} = \mathbb{R}^d$, we could define the contexts that comprise **C** over D in the following way: let $C_{YELP} = (V_{YELP}, f_{YELP})$, where V_{YELP} is all the terms that occur in the Yelp business reviews and $f_{YELP}(w)$ yields a d-dimensional word vector for w learned on the Yelp dataset; similarly let C_{IMDB} have V_{IMDB} as all of the terms that occur in IMDB and $f_{IMDB}(w)$ yield a d-dimensional word vector for w learned on IMDB; C_{POS} and C_{NEG} can be defined similarly, with vocabularies and word vectors coming from only the positive and negative reviews across both datasets, respectively. We could add a background context C_{ALL} with a vocabulary consisting of all terms used across both datasets and with word vectors learned on the union of both datasets.

The comparison operator Φ takes multiple contexts and outputs a list of (word, score) tuples for each term in the shared vocabulary:

$$\Phi(C_j, \ldots, C_k) = \left\langle (w, \phi(w, C_j, \ldots, C_k) \mid w \in \bigcap_{i=j}^{k} V_i \right\rangle$$

where the scoring function ϕ is user-defined and task-specific. For example, if our task is to identify words used similarly across contexts, our scoring function can be specified to give high scores to terms whose usage is similar across the contexts.

The scored terms returned from Φ are able to be processed by operators such as `head` (return the highest-scored terms), `tail` (return the lowest-scored terms), and `average` (return the average scores of all the terms).

As an example application, we can use disjoint temporal segments as our context views in a term-

[2] `https://www.yelp.com/dataset_challenge`

focused task. Let C_1 be the initial time period context and let C_2 be the final time period context. We wish to discover a list of $w \in V_1 \cap V_2$ that underwent semantic change. We define a ϕ such that a given term with similar annotations across C_1 and C_2 will have a *higher* score, and a given term with different annotations across C_1 and C_2 will have a *lower* score. Thus, when we run $\texttt{head}(\Phi(C_1, C_2))$ the result is the terms that changed the least while $\texttt{tail}(\Phi(C_1, C_2))$ will show the terms that changed the most, i.e., underwent semantic change. We discuss this particular application scenario in more depth in the next section.

6.3 Analysis of Semantic Change

The evolution of word usage is a well-studied area in linguistics. Also known as *semantic change* or *diachronic analysis*, it has received attention in the NLP community, most recently by Kim et al. (2014), Kulkarni et al. (2015), and Hamilton et al. (2016). All three methods are based on word embedding similarity, and learn separate embeddings for distinct time periods. For a brief outline of each method, see section 6.7. With these techniques, we can discover how words such as *awful* change meaning over time. In the 1850s, it meant *solemn* or *majestic*, whereas in the 1900s it meant *terrible* or *horrible* (Hamilton et al., 2016). Detecting and analyzing these semantic shifts allows us to learn about the culture and evolution of language.

We next formalize the problem in the CCLA framework and compare our findings to previous results.

6.3.1 CCLA Formulation

In this task, we will use disjoint temporal segments as our context views in a term-focused task. Let C_1 be the initial time period context and let C_2 be the final time period context. We wish to discover $w \in V_1 \cap V_2$ that underwent semantic change.

We define the following scoring function:

$$\phi(w, C_1, C_2) = \cos(NN(w, C_1),\ NN(w, C_2))$$

where NN finds the top-k nearest neighbors of w in C_i (and their corresponding similarities) by using its d-dimensional word vector annotation $f_i(w) \in \mathbb{R}^d$. Since the word vectors are normalized to unit length, the nearest neighbors are calculated using a dot product against all other word vectors in each embedding space.

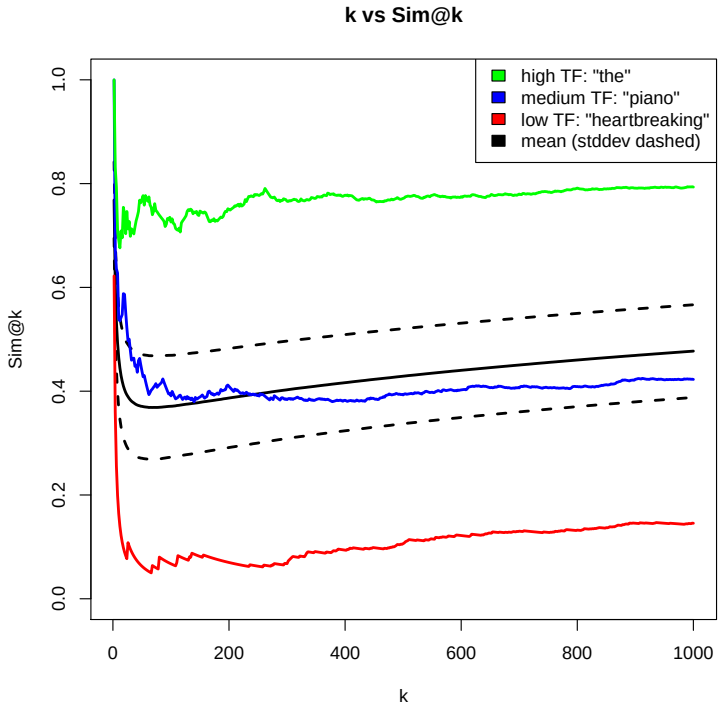

Figure 6.1: The effect of the k parameter in cosine similarity of top-k similarity scores.

Consider the following example of $\phi(w, C_1, C_2)$ for $w = sour$ and $k = 3$:

$$NN(w, C_1) = \langle(\text{grapes}, 0.47), (\text{sweet}, 0.40), (\text{meek}, 0.38)\rangle$$

$$NN(w, C_2) = \langle(\text{bitter}, 0.41), (\text{tart}, 0.39), (\text{sweet}, 0.37)\rangle$$

To compute the cosine similarity of these two nearest-neighbor lists, we have the following:

$$\cos(NN(w, C_1),\ NN(w, C_2)) = \frac{0.40 \cdot 0.37}{\sqrt{0.47^2 + 0.40^2 + 0.38^2} \cdot \sqrt{0.41^2 + 0.39^2 + 0.37^2}} = 0.302$$

That is, we take a *dot product over the shared vocabulary* with the nearest-neighbor similarities as weights and then divide by the product of each list's magnitude. This results in a maximum score of 1.0 when all k dimensions are exactly the same and a minimum score of 0.0 when none of the dimensions in the top-k match. Essentially, ϕ measures how similar the usage of a particular w is across the two contexts.

Figure 6.1 shows the effect of k on the similarity scores. If we have very small k values, the similarity is very high, since it's always the case that the top-ranked term is the term itself. There are usually 1 or 2 other terms that are highly correlated, which gives high scores. Then, as we increase k past about 4,

the similarity sharply drops as the different contexts begin to come into play. Lastly, we slowly increase the similarity score as more and more terms are added, since there is a greater and greater chance of matching dimensions.

In black, we show the average similarity score for the IMDB corpus for positive versus negative contexts. The dashed lines show the boundary of one standard deviation in similarity. If we plot individual words (in green, blue, and red), we can see they are relatively stable with respect to themselves. The individual words displayed have high TF, medium TF, and low TF. This is to show that (as one would expect) the higher the TF, the easier it is to match the top-k terms. In conclusion, Figure 6.1 shows that the similarity scores are relatively robust to the setting of k.

The function Φ's output is a list of unsorted (term, score) tuples. From the current example, this could look like

$$\Phi(C_1, C_2) = \langle (\text{sour}, 0.302), (\text{plane}, 0.122), \ldots, (\text{the}, 0.506) \rangle$$

Thus, to find words whose usage changed the most (i.e., underwent semantic change), we find the w's with the least similar usage: $\texttt{tail}(\Phi(C_1, C_2))$, which returns the tuples with the lowest scores. To find the most stable words (i.e., those whose meaning changed the least), we would instead use `head`.

6.3.2 Experiments

We compare our method to Hamilton et al. (2016) and use the COHA corpus (Davies, 2010) to contrast word usage in English fiction between $C_1 = 1900$ and $C_2 = 1990$. For word annotations, we used PPMI, SVD, and SGNS (skipgram with negative sampling from Mikolov et al. (2013b)) word vectors released by Hamilton et al. (2016). We set $k = 500$ in the nearest-neighbor scoring function to capture a fair amount of similar words while reducing noise farther down in the neighbor lists.

Table 6.1 compares the results using the CCLA framework with the semantic change detection described in Hamilton et al. (2016). As with the previous work, we found SVD and SGNS to outperform PPMI. Interestingly, SVD appears to be slightly ahead of SGNS, in contrast to the previous results. Despite this, it has been shown that SVD may be superior to SGNS in some evaluation cases (Levy et al., 2015). Some detected words are shared with those found in Hamilton et al. (*headed*, *gay*) and some words were detected by multiple methods with CCLA (*figured*, *gay*, *handling*, *compound*).

Table 6.2 shows the nearest-neighbor lists for the words detected to have changed the most by SVD and SGNS. We see that *plane* shifted from meaning a type of inclined or flat surface to a shortened form of *airplane*. The term *figured* changed meaning from describing one's figure (body) to an act of making a decision.

Words that changed the least (i.e., were the most similar) from 1900 to 1990 were non-content words

PPMI		SVD		SGNS	
Hamilton et al.	CCLA	Hamilton et al.	CCLA	Hamilton et al.	CCLA
know	**gay**	harry	**handling**	**wanting**	**figured**
got	favorite	**headed**	**plane**	**gay**	**guy**
would	arrangement	**calls**	**headed**	**check**	random
decided	please	**gay**	**gay**	**starting**	**gay**
think	which	whenever	**figured**	**major**	**chick**
stop	**handling**	male	**compound**	**actually**	**compound**
remember	random	**actually**	**kid**	touching	**notices**
started	distributed	special	random	harry	checking
must	available	cover	**reverse**	**headed**	perspective
wanted	otherwise	naturally	division	romance	**handling**

Table 6.1: Comparing methods to find the most-changed words between 1900 and 1990. Each method operates on a type of word representation (PPMI, SVD, or SGNS). We follow the conventions of Hamilton et al. (2016) in bolding terms the authors agree to be clearly correct after consulting a dictionary, underlining borderline cases, and leaving incorrect terms unmarked.

Word	Vector	Nearest-neighbors in 1900s	Nearest-neighbors in 1990s
handling	SVD	ribbon, threads, buttons, silk, yellow	delivery, enclosed, send, additional, tax
plane	SVD	level, higher, above, horizon, beneath	train, pilot, engines, jet, flight
figured	SGNS	thread, lace, rip, lined, stockings	figure, find, thought, pointed, remember
guy	SGNS	jane, grey, thomas, chester, roger	tough, person, kid, fellow, man

Table 6.2: Example words that changed dramatically during the 20th century. The examples were chosen from the top-20 most-changed lists from words in Table 6.1.

IMDB			Yelp		
Positive	Negative	Ambiguous	Positive	Negative	Ambiguous
shift	travesty	loggia	trails	zero	behavior
magnificent	drivel	krigie	gem	tasteless	planner
lovingly	utter	morton	heavenly	edible	advertising
observed	inane	clifford	freshest	flavorless	reaching
heartbreaking	pile	griffin	wonderfully	apology	silverware
determination	abysmal	chad	gifts	shitty	arrogant
marvelous	unfunny	epps	tastings	lousy	avoiding
tightly	idiotic	deluise	hike	irritated	gratuity
globe	nonsensical	perkins	scrumptious	clerk	collections
superbly	wretched	ana	explore	error	cc

Table 6.3: Using embedding annotation similarity to discover the top positive, negative, and ambiguous terms for IMDB and Yelp. Each corpus treated independently as a separate CCLA problem.

such as *never*, *not*, *eight*, *six*, and *twenty*. These are produced when using $\texttt{head}(\Phi(C_1, C_2))$.

6.4 Comparative Lexical Analysis over Context

A context-aware lexical analysis allows us to discover both context-sensitive and context-insensitive terms. Context-sensitive terms are those that may be used to represent their respective context. For example, *excellent* and *great* could represent a positive sentiment context and *bad* and *horrible* could represent negative sentiment contexts. Context-*in*sensitive terms are those that do not change across contexts, such as stop words. Intelligently assigning scores to these word types will allow us to rank words per context, and even allow us to discover ambiguous words (those whose meaning changes between contexts). Topic models have been used to address some of these issues, and we discuss their differences and limitations in more depth in section 6.7. Tan et al. (2015) investigated finding ambiguous terms between two corpora, but not in a general contextual text mining framework. In the next sections, we will show how to address these goals with CCLA.

6.4.1 CCLA Formulation

First, we will find ambiguous—or, "context-sensitive"— words between two disjoint contexts in a score-focused manner. We ask the following question: which words' surroundings change the most between C_1 and C_2? In semantic change detection, C_1 and C_2 were time periods. Here, we will use contexts from the same time, but with different metadata attributes. Concretely, imagine D is a sentiment analysis dataset. If we let $C_1 = (V_1, f_1)$ where V_1 is the set of all words used in positive documents and $f_1(w)$ is a d-dimensional word vector learned from only the positive documents, and similarly for C_2 with the

Word	Sentiment	Phrases
loggia	Positive	**loggia** is wonderful as tony's boss, lopez watching o'connor and **loggia** [...] is pure poetry **loggia** i always enjoyed watching in just seeing him yell
	Negative	**loggia** played his character so lamely, you didn't care **loggia** is about as heroic as a bored businessman. ridiculous attempt at a hispanic accent. (sorry **loggia**.)
krige	Positive	great cast with alice **krige** and brian krause alice **krige** plays the borg queen again fantastically played by beautiful and talented alice **krige**.
	Negative	usually excellent alice **krige** is wasted in this one alice **krige** seems to shoulder the film, **krige** gave the only convincing performance
morton	Positive	love's rebound with socialite anne **morton** (ruth roman) cannavale, rory culkin, joe **morton**, sandra oh, john and **morton** selden (as oberon's grandfather)
	Negative	thriller from directors rocky **morton** and annabel **morton** is too strong an actress to be relegated is a poor replacement for bob **morton**'s charismatic
clifford	Positive	is a fledgling playwright named **clifford** anderson old student named **clifford** anderson (christopher reeve) impressed with christopher reeve as **clifford** anderson.
	Negative	dumps louque for his mate **clifford** grayson her love for his pal, **clifford** greyson (robert noland) his companion **clifford** grayson. what a yawn-fest
griffin	Positive	and co-stars **griffin** dunn ('after hours') send his younger brother (**griffin** dunne) to law school **griffin** dunne is very well cast as the man
	Negative	how did lee and **griffin** become such deep friends put together by the hack **griffin** jay who wrote his awful behavior, peter **griffin** has no excuse.

Table 6.4: Usage samples of the five most ambiguous words (all actors and actresses) in the IMDB dataset. In some cases, the same person is discussed in different ways; in others, people share the same name, leading to ambiguity.

negative documents, we can discover (1) which words are the most stable between sentiments and (2) which words change the most (i.e., are ambiguous) between sentiments.

We will use the exact same ϕ as in section 6.3:

$$\phi(w, C_1, C_2) = \cos(NN(w, C_1), NN(w, C_2))$$

Now, using $\texttt{head}(\Phi(C_1, C_2))$ we retrieve stable words between sentiment polarities and using `tail` we discover ambiguous words.

Second, we want to find words that are representative of their context. In the sentiment analysis example, we hope to find words like *amazing* in C_1 and *terrible* in C_2. To accomplish this, we design

Shared between IMDB and Yelp		
Positive	Negative	Ambiguous
magnificent	unfunny	unparalleled
marvelous	incoherent	panoramic
breathtaking	unimaginative	unmatched
heavenly	inane	daunting
understated	abysmal	tantalizing
splendid	horrid	aligned
exquisite	moronic	soft
timeless	atrocious	hardworking
inspirational	nonsensical	descriptive
delectable	idiotic	serene

Positive			Negative		
IMDB	Yelp	Ambiguous	IMDB	Yelp	Ambiguous
disturbing	helpful	orthodox	corny	watery	overzealous
effective	whipped	gargantuan	contrived	rubbery	ravenous
political	flaky	accented	unbelievable	polite	functional
engaging	polite	mirrored	unoriginal	surly	desolate
brutal	generous	copious	convincing	oily	impersonal
dramatic	fluffy	pungent	wealthy	mushy	squashed
touching	prompt	textured	inept	sticky	callous
powerful	quaint	sweltering	graphic	helpful	grubby
striking	trendy	conscientious	scary	drenched	sturdy
shocking	efficient	kooky	predictable	crusty	blah

Table 6.5: Using embedding annotations to compare term contexts between IMDB and Yelp. Cross-corpus lists show words that are used similarly in both collections. Corpus-specific lists show words that are used differently given a particular collection.

a second scoring function Φ' which uses the previous Φ. We include a third "background" context C_B that covers all the documents in D. To find representative words in C_1 (i.e., positive words), we use $\texttt{head}(\Phi'(C_1, C_B))$, where

$$\phi'(w, C_1, C_B) = \phi(w, C_1, C_B) - \phi(w, C_1, C_2).$$

The first term compares word contexts in C_1 with the background. Recall that ϕ gives a high score if the word shares similar neighbors and a low score if the word has different neighbors. A high score may result from two situations: (1) the word's usage is the same in both contexts (e.g., a stop word), or (2) the word's usage is primarily in C_1, so when combined with C_B, its usage doesn't change.

To filter out the stop words from $\phi(C_1, C_B)$ we subtract $\phi(C_1, C_2)$, since the second term assigns high scores to stable words—stop words. This leaves terms that represent C_1 well. Naturally, the same may be done to find words specific to C_2.

6.4.2 Experiments

We perform a few different experiments on two popular sentiment analysis datasets, IMDB movie reviews and the Yelp academic dataset. For all experiments, we used 300-dimensional word vectors as term annotations that were learned by GloVe (Global Vectors by Pennington et al. (2014)) with the following untuned parameters: window size = 15, max iterations = 25, and a minimum term count of 10 in each corpus.

Table 6.3 shows two separate CCLA experiments. In each case, we set C_1 = terms from positive documents, C_2 = terms from negative documents, and C_B = terms from the entire dataset. As in section 6.3, we set $k = 500$ for the nearest-neighbor lists. We use Φ to discover ambiguous words and Φ' to find representative words.

As expected, words such as *travesty* describe negative tones: "a hopelessly miscast, misdirected travesty of actors." At first glance, *shift* may seem a strange choice for positive feelings, but when examined in context, it makes sense: "display her native rhythm and ability to shift tempo in the lavish production" and "a 180-degree shift from the idealistic rhetoric portrayed in [other] offerings."

Ambiguous words offer hints at sentiment targets. In IMDB, the most ambiguous terms are all names of actors and actresses. Table 6.4 shows example sentences where these words are used. In Yelp, the ambiguous terms are more varied; staff *behavior* could be good or bad and *silverware* could be clean or dirty. Credit cards ("CC") may or may not be accepted.

Table 6.5 compares different context views that span both IMDB and Yelp. The "Shared" row sets C_1 = all words in positive documents, C_2 = all words in negative documents, and C_B = words across all documents in both datasets.

The "Pos only" and "Neg only" rows split C_i by positive and negative documents across both corpora. Ambiguous words in these two rows refer to distinguishing terms between all positive documents based on the corpus or all negative documents based on the corpus.

For example, we can learn the following from this analysis:

1. *magnificent* is used similarly in both datasets for positive sentiment;
2. *unparalleled* is used differently in terms of positive and negative sentiment in both datasets;
3. *disturbing* can be a positive word to describe movies[3], but is not a positive way to describe businesses;
4. *helpful* can be a positive word to describe businesses, but is not a positive way to describe movies;
5. *overzealous* is a negative word in both datasets, but used differently in IMDB vs. Yelp.

[3] "This movie is both disturbing and extremely deep" / "...very compelling, even disturbing, a chill ran down my spine."

When comparing across corpora, we have the issue of disjoint vocabulary. For example, "movie" is used much more in IMDB reviews than Yelp, even though the term occurs in both. Thus, when comparing positive reviews, "movie" will seem like it's a positive word for IMDB. To combat this, we filter the lists from each cross-corpus analysis, only keeping adjectives.

Since each word is scored with respect to its context, it is a natural extension to use these scored terms in feature selection or even to estimate word polarity scores. Further, scoring terms based on sensitivity to different contexts can be very useful for domain adaptation and transfer learning since we can treat both the source domain(s) and the target domain as contexts to identify terms semantically "stable" across domains, which are intuitively more generalizable than terms very sensitive to domain variations. We would expect shared positive and negative terms between IMDB and Yelp to aid in other sentiment analysis tasks, where the corpus-specific terms are less helpful. The fact that this works even when there is no labeled data in the target domain results in a completely *unsupervised* way to received specialized knowledge.

6.5 Comparing Word Annotations

It is educational to study how annotations drawn from the same data are similar or different. There are many ways to compare embedding methods as annotations using downstream tasks like word analogies or word similarity scoring (Levy et al., 2015). But is there a way to explicitly compare the structure learned by these models? If we have a quantification of this structure, does it give any information about task performance? Levy et al. (2015) consider different word embedding parameters such as adding context vectors (GloVe and SGNS), eigenvalue weighting (SVD), and vector normalization. Other configurations mentioned (but not tested) are number of iterations, vector dimensionality, and effect of randomness.

As a demonstration of CCLA's flexibility in choice of context definition, we explore the concept of *word embedding stability*. We define word embedding stability as a measure of how consistent nearest-neighbor lists are across different runs of the same algorithm. Consistency is an important attribute when replicating results or comparing two methods against one another. Different random seeds may play some role in the quality of the word vectors, and methods that use random sampling (like SGNS) may be affected. Nearest-neighbor lists are critical when solving word analogy problems or measuring the similarity between words, so this is the aspect of the word vectors that we will consider while measuring stability.

6.5.1 CCLA Formulation

In sections 6.3 and 6.4, we varied the vocabularies for each context. Now, we will vary the word annotations instead in annotation-focused experiments.

Let C_1 and C_2 represent the same text data (and hence $V_1 = V_2$), but define $f_1(w)$ and $f_2(w)$ as yielding word vectors learned by the same word embedding method with a *different random initialization*. We wish to measure how similar the embeddings are for different runs of the same algorithm.

In the CCLA framework, one way to address this situation requires a similarity metric to measure the nearest-neighbors of the two runs. Before, we used cosine similarity with the term annotation dot product scores as term weights. If we want to stress the orders of the lists themselves, we should ignore the weights and use a ranking correlation metric. The flexibility of CCLA allows us to choose the best measure to suit our task. A rank difference near the top of the lists should be more detrimental than a rank difference farther down the list. In other words, heavy bias should be placed on getting similar top terms to match, rather than terms farther down the list. For this reason, we choose normalized discounted cumulative gain (NDCG) as our measure. Discounted cumulative gain is defined as

$$DCG@n = \sum_{i=1}^{n} \frac{r_i}{\log_2(i+1)}$$

where each element at rank i has a relevance score r_i. Normalized DCG divides DCG by the ideal ranking, i.e. sorting the top n elements by their decreasing relevance and taking their DCG.

To measure embedding stability, we consider the two ranked nearest-neighbor lists for w from C_1 and C_2. We call C_1's list the ideal ranking and assign the relevance scores $n, n-1, \ldots, 1$ to the top n items. We then measure NDCG of C_2 with respect to C_1's neighbors as a rank correlation metric, defining the function NDCG@$n(w, C_1, C_2)$. Therefore, stable methods will have a higher average NDCG@n than less stable methods. For our application, NDCG@n is the following.

$$NDCG@n = \frac{DCG@n}{iDCG@n} = \frac{DCG@n}{\sum_{i=1}^{n} \frac{n-i+1}{\log_2(i+1)}}$$

Note that in our case, the ideal DCG is always the same for each value of n, since we call the top-ranked word the most relevant, the second-ranked word the second-most relevant, and so on.

We can now state ϕ for embedding stability measurement as

$$\phi(w, C_1, C_2) = \text{NDCG@}n(w, C_1, C_2)$$

and overall stability score $\texttt{average}(\Phi(C_1, C_2))$.

	SGNS		GloVe	
Dataset (task)	Low	High	Low	High
Google (word analogies)	38.70	**44.74**	11.98	**26.52**
MSR (word analogies)	53.41	**56.32**	13.92	**31.92**
MEN (word similarity)	51.85	**58.06**	23.80	**38.24**
Rare (word similarity)	44.37	**58.09**	29.18	**42.92**

Table 6.6: We compare the effect of stability (low vs. high) using analogy and word similarity benchmarks. While stability does not seem to indicate performance differences across embedding methods, it does suggest that a higher stability indicates higher performance within-method.

As an example, we compute the NDCG@5 for the term *sour*. Imagine the top-k terms for C_1 are $\langle A, B, C, D, E\rangle$ and the top-k terms for C_2 are $\langle B, A, F, D, C\rangle$. We take the list returned by C_1 as the ideal ranking by assigning the relevance scores $\langle 5, 4, 3, 2, 1\rangle$. This gives an ideal DCG of

$$\frac{5}{\log_2(2)} + \frac{4}{\log_2(3)} + \frac{3}{\log_2(4)} + \frac{2}{\log_2(5)} + \frac{1}{\log_2(6)} = 10.27192.$$

For C_2's list, we would get the relevance scores $\langle 4, 5, 0, 2, 3\rangle$ for a DCG of

$$\frac{4}{\log_2(2)} + \frac{5}{\log_2(3)} + \frac{0}{\log_2(4)} + \frac{2}{\log_2(5)} + \frac{3}{\log_2(6)} = 9.17656.$$

Thus, the NDCG@5 for the term *sour* would be $\frac{9.17656}{10.27192} = 0.8933637$. If we had a small $|V| = 3$ where

$$\Phi(C_1, C_2) = \langle (\text{sour}, 0.89), (\text{plane}, 0.74), (\text{the}, 0.82)\rangle,$$

then $\texttt{average}(\Phi(C_1, C_2)) = 0.8166667$.

Because NDCG@$n(w, C_1, C_2)$ is not necessarily equal to NDCG@$n(w, C_2, C_1)$, it's important to take multiple measurements to understand the consistency of the score. In our case, we take several measurements over different random seeds for each comparison and report results with standard deviations for interpretability (see Figure 6.2, to be discussed later).

Note that we can use this framework to compare embeddings not only from different seeds, but from different algorithms or even dimensions. This measure could be used to see how similarly two or more algorithms perform on the same data.

6.5.2 Experiments

We use public word embedding implementations to measure the stability of both GloVe (Massung et al., 2016) and SGNS[4] at various numbers of iterations and test whether stability may be an indicator of task

[4] `https://bitbucket.org/yoavgo/word2vecf`

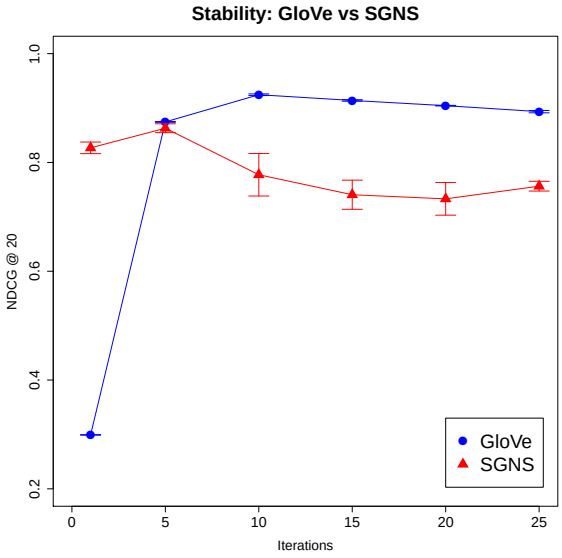

Figure 6.2: Using NDCG of nearest-neighbor lists to measure the stability of GloVe and SGNS by iteration.

performance. We used 300-dimensional embeddings trained on the IMDB dataset. In both cases, we used a symmetric window of size 8 with the remaining parameters set to their defaults. For the NDCG measure, we set $n = 20$ to stress performance at the top of the nearest-neighbor lists.

Figure 6.2 shows the CCLA stability scores from 1 to 25 iterations. Each point on the chart is the average of 10 different random seeds with error bars denoting the standard deviation of the stability scores. SGNS is initially stable, but starts to drop as iterations increase, perhaps indicative of overfitting or model divergence. GloVe's word vectors are fairly consistent after 10 iterations.

We used standard benchmarks for word analogy solving and word similarity scoring. Google analogies (Mikolov et al., 2013a) and MSR analogies (Mikolov et al., 2013c) are written in the form "a is to b as c is to d" (where d must be determined). The MEN (Bruni et al., 2012) and Rare (Luong et al., 2013) word similarity tests present word pairs with human-assigned similarity scores. This task is evaluated by measuring the embedding similarity scores' correlation with human judgements via Spearman's ρ.

Table 6.6 compares task performance on embeddings with low stability vs. high stability. For SGNS, we used iteration 25 as the low stability point and iteration 5 as the high stability point; for GloVe, we used 5 as low and 10 as high. SGNS outperformed GloVe in all tasks, even at low stability. Thus, comparing stability across methods may not be a viable metric at suggested performance. Despite this, looking *within*-method, CCLA's stability measure does seem to indicate that lower-stability runs do underperform the higher-stability runs. This is an especially interesting result for SGNS, since the high stability point is actually at a much lower number of iterations. This suggests that we might use stability as an early-

stopping criterion when learning the word representations, potentially saving much compute time while increasing performance.

6.6 CCLA for Text Representation

In sections 6.3, 6.4, and 6.5, we focused on how CCLA can be used to explain differences across corpora or annotations. In each section, we briefly touched on how CCLA can be used to create feature representations themselves. In this section, we explicitly list a few scenarios where CCLA can be used for explanatory text representation with attention on downstream task applicability.

Feature selection for machine learning problems follows directly from comparative lexical analysis over context. The top words for class C_1 selected by $\texttt{head}(\Phi'(C_1, C_B))$ provide a principled way to select the terms indicative of C_1, a one-sided metric (Zheng et al., 2004). Negative features, i.e., those that create a two-sided metric are not explicitly modeled in the previous formula. However, we can find the context-sensitive features (ambiguous features between class labels) via $\texttt{tail}\Phi'(C_1, C_B)$ and *not* use them.

Depending on the definition of our initial scoring function, Φ gives a real score per term with a possible bound. In section 6.4 we used

$$\phi(w, C_1, C_2) = \cos(NN(w, C_1), NN(w, C_1))$$

and

$$\phi'(w, C_1, C_B) = \phi(w, C_1, C_B) - \phi(w, C_1, C_2).$$

Note that since we use cosine similarity in positive space such that $\phi \in [0, 1]$. This results in $\phi' \in [-1, 1]$. Since we have this convenient bound, the per-term score ϕ' can easily be included in a linear feature selection model.

For example, if we wished to include terms with a higher document frequency (df, the fraction of documents a particular term appears in), we could rewrite our feature scoring function for term w with respect to context C_1 as

$$score(w, C_1) = \phi'(w, C_1, C_B)^{\beta_1} + df^{\beta_2}$$

where β_1 and β_2 are parameters that can be set or learned depending on the corpus or task.

Transfer learning is also a potential application for lexical CCLA scores. As an example, imagine that we have a sentiment analysis problem to perform on some previously unknown, streaming data. We would like to select robust sentiment features that are applicable in a wide range of domains. As

we showed in section 6.4, there are some particular terms that only carry meaningful sentiment when considered in certain contexts. Our goal would be to use CCLA to select those generalizable terms that are able to explain sentiment. In other words, we wish to find context-insensitive (i.e., unambiguous with regard to polarity) terms across sentiment analysis corpora.

Consider the case where we have two datasets, C_1 and C_2. We wish to find terms that are positive in both and negative in both. We can use the same setup as the feature selection task, where we use positive C_1 and C_2 documents as one comparison and negative C_1 and C_2 documents as another. Further, we can explicitly drop the context-sensitive terms found between the two corpora to reduce the dataset bias on certain terms when applied to our streaming application.

Such a use of CCLA allows practitioners to explain why certain terms were selected: *awful* was chosen because it is a clearly negative term across domains and *soft* was explicitly dropped because it is not clear what its sentiment is given our existing cross-domain corpora.

Selecting word embeddings to use in a task is a choice researchers often face when designing an end-to-end NLP or machine learning system. Word embeddings play a crucial role since their quality significantly determines the overall performance of the system (Collobert et al., 2011). Usually, we are able to test the system with multiple word embeddings and gauge the performance. However, it is not always the case that we have labeled training data or have the time to run many experiments, which is especially true if the word embeddings are input to a neural network architecture.

In this case, we can design CCLA functions that measure some property that we wish our word annotations (i.e., embeddings) to capture. Due to the flexible nature of CCLA, we can even incorporate multiple measurements into our score. For example, we may desire word embeddings that are stable (see section 6.5) while also attaining high scores in word similarity tasks. This strategy is not limited to word embeddings in particular, and can be applied to any word annotation. Performing this type of "annotation selection" is quite similar to feature selection.

6.7 Related Work

Our work spans several areas of research:

Detecting semantic change. Hamilton et al. (2016) suggest orthogonal Procrustes to align word embedding spaces learned from different time periods, in contrast to per-word heuristics for the alignment Kulkarni et al. (2015). Kim et al. (2014) start at time period t and learn embeddings. They initialize time period $t+1$ with those from t, and measure which words' cosine similarities changed the most. Unlike the previous two works, this does not produce a mapping function. We propose an approach that does not require embedding matrix alignment and thus does not require an optimization algorithm; we

utilize within-period word similarities to create word representations that are comparable across time. This also removes the constraint of incrementally retraining the embeddings each time step; instead of learning 10 embeddings to compare between t_1 and t_{10}, we learn two and directly compare them with CCLA.

Contextual text mining. Topic models have been extended to support analysis of topic variations over different contexts in many ways. In CPLSA (Mei and Zhai, 2006), a generalized form of Zhai et al. (2004), context is incorporated into a topic model as explicit variables. A flexible way to incorporate arbitrary features into a topic model, Dirichlet-multinomial regression, was proposed by Mimno and McCallum (2008). Related recent work is the differential topic model (Chen et al., 2015). There are many topic models for supporting topic analysis in association with specific context such as time and location (e.g., Mei et al. (2006); Yuan et al. (2013)). A common idea in all these and other methods is to model the association of context and topics as word distributions, facilitating cross-context *topic* analysis, but cannot easily support cross-context *lexical analysis*, which is our main goal. An important difference between our work and these contextual topic models is that our approach does not make parametric assumptions in modeling text (which are generally needed in topic models) and is very flexible, allowing it to easily work with any context and context-specific word annotations.

Word embedding evaluation. Word embeddings like SGNS (Mikolov et al., 2013b)) and GloVe (Pennington et al., 2014) have become standard repertoire in text mining and NLP. Some work has been done examining the methods and parameters themselves (Levy and Goldberg, 2014; Levy et al., 2015). Faruqui et al. (2016) find issues with using word similarity as evaluation for embeddings, and suggest only to consider downstream task performance. Our method is able to compare the embedding spaces themselves, which may be a useful alternative to premade similarity datasets or the less direct application tasks.

6.8 Conclusions and Future Work

We propose a general way to perform cross-context lexical analysis to accommodate *any* notion of context, *any* similarity function, and *any* type of word annotation. This enables many new applications all under the same framework (e.g. development of a common toolkit to support all applications), including analysis of semantic change, comparative analysis of meaning over context, and word embedding stability evaluation.

CCLA opens up interesting new directions for further study, especially in additional applications. One use is to investigate framing bias on political viewpoints. Another is a more fine-grained comparative analysis over specific products as opposed to movies or businesses. Term scoring can be further taken

advantage of in sentiment valence prediction. Pablos et al. (2016) use word vector similarity to create sentiment valence scores per term, but they only consider similarity with a manually-chosen positive and negative word. Word sense disambiguation is another unvisited technique, and CCLA's notion of context could help determine which words have multiple senses. Using CCLA as a tool in a larger system is desirable, such as learning to automatically partition a corpus to maximize word differences, or using it for event detection when tones shift from a monitored stream. We want to investigate embedding comparisons further using larger training data and automatically determine an optimal dimensionality or window size given new scoring functions.

CHAPTER 7

APPLICATION SYNTHESIS: META

This chapter shows how the previous work has been integrated into free, open-source tools available online. This enables broad impact through reproducibility and enables faster discovery of future work.

META is developed to unite machine learning, information retrieval, and natural language processing in one easy-to-use toolkit. Its focus on indexing allows it to perform well on large datasets, supporting online classification and other out-of-core algorithms. META's liberal open source license encourages contributions, and its extensive online documentation, forum, and tutorials make this process straightforward. We run experiments and show META's performance is competitive with or better than existing software.

7.1 A Unified Framework

As NLP techniques become more and more mature, we have great opportunities to use them to develop and support many applications, such as search engines, classifiers, and integrative applications that involve multiple components. It's possible to develop each application from scratch, but it's much more efficient to have a general toolkit that supports multiple application types.

Existing tools tend to specialize on one particular area, and as such there is a wide variety of tools one must sample when performing different data science tasks. For text-mining tasks, this is even more apparent; it is extremely difficult (if not impossible) to find tools that support both traditional information retrieval tasks (like tokenization, indexing, and search) alongside traditional machine learning tasks (like document classification, regression, and topic modeling).

Table 7.1 compares META's many features across various dimensions. Note that only META satisfies all the areas while other toolkits focus on a particular area. In the case where the desired functionality is scattered, data science students, researchers, and practitioners must find the appropriate software packages for their needs and compile and configure each appropriate tool. Then, there is the problem of data formatting—it is unlikely that the tools all have standardized upon a single input format, so a certain amount of "data munging" is required. All of this detracts from the actual task at hand, which

	Indri *IR*	**Lucene** *IR*	**MALLET** *ML/NLP*	**LIBLINEAR** *ML*	**SVM**^{MULT} *ML*	**scikit** *ML/NLP*	**CoreNLP** *ML/NLP*	**META** *all*
Feature generation	✓	✓	✓			✓	✓	✓
Search	✓	✓						✓
Classification			✓	✓	✓	✓	✓	✓
Regression			✓	✓	✓	✓	✓	✓
POS tagging			✓				✓	✓
Parsing							✓	✓
Topic models			✓			✓		✓
n-gram LM								✓
Word embeddings			✓				✓	✓
Graph algorithms								✓
Multithreading		✓	✓			✓	✓	✓

Table 7.1: Toolkit feature comparison. Citations for all toolkits may be found in their respective comparison sections.

has a marked impact on productivity.

The goal of the META project is to address these issues. In particular, we provide a unifying framework for existing machine learning and natural language processing algorithms, allowing researchers to quickly run controlled experiments. We have modularized the feature generation, instance representation, data storage formats, and algorithm implementations; this allows users to make seamless transitions along any of these dimensions with minimal effort. Finally, META is dual-licensed under the University of Illinois/NCSA Open Source Licence and the MIT License to reach the broadest audience possible.

Due to space constraints, in this paper, we only delve into META's natural language processing (NLP), information retrieval (IR), and machine learning (ML) components in section 7.3. However, we briefly outline all of its components here:

Feature generation. META has a collection of tokenizers, filters, and analyzers that convert raw text into a feature representation. Basic features are *n*-gram words, but other analyzers make use of different parts of the toolkit, such as POS tag *n*-grams and parse tree features. An arbitrary number of feature representations may be combined; for example, a document could be represented as unigram words, bigram POS tags, and parse tree rewrite rules. Users can easily add their own feature types as well, such as sentence length distribution in a document.

Search. The META search engine can store document feature vectors in an inverted index and score them with respect to a query. Rankers include vector space models such as Okapi BM25 (Robertson et al., 1994) and probabilistic models like Dirichlet prior smoothing (Zhai and Lafferty, 2004). A search demo is online[1].

Classification. META includes a normalized adaptive stochastic gradient descent (SGD) implementation (Ross et al., 2013) with pluggable loss functions, allowing creation of an SVM classifier (among

[1] `https://meta-toolkit.org/search-demo.html`

others). Both ℓ_1 (Tsuruoka et al., 2009) and ℓ_2 regularization are supported. Ensemble methods for binary classifiers allow multiclass classification. Other classifiers like naïve Bayes and k-nearest neighbors also exist. A confusion matrix class and significance testing framework allow evaluation and comparison of different methods and feature representations.

Regression. Regression via SGD predicts real-valued responses from featurized documents. Evaluation metrics such as mean squared error and R^2 score allow model comparison.

POS tagging. META contains a linear-chain conditional random field for POS tagging and chunking applications, learned using ℓ_2 regularized SGD (Sutton and McCallum, 2012). It also contains an efficient greedy averaged perceptron tagger (Collins, 2002).

Parsing. A fast shift-reduce constituency parser using generalized averaged perceptron (Zhu et al., 2013) is META's grammatical parser. Parse tree featurizers implement different types of structural tree representations (Massung et al., 2013). An NLP demo online presents tokenization, POS-tagging, and parsing[2].

Topic models. META can learn topic models over any feature representation using collapsed variational Bayes (Asuncion et al., 2009), collapsed Gibbs sampling (Griffiths and Steyvers, 2004), stochastic collapsed variational Bayes (Foulds et al., 2013), or approximate distributed LDA (Newman et al., 2009).

***n*-gram language models** (LMs). META takes an ARPA-formatted input[3] and creates a language model that can be queried for token sequence probabilities or used in downstream applications like SyntacticDiff (Massung and Zhai, 2015).

Word embeddings. The GloVe algorithm (Pennington et al., 2014) is implemented in a streaming framework and also features an interactive semantic relationship demo. Word vectors can be used in other applications as part of the META API.

Graph algorithms. Directed and undirected graph implementations exist and various algorithms such as betweenness centrality, PageRank, and myopic search are available. Random graph generation models like Watts-Strogatz and preferential attachment exist. For these algorithms see Easley and Kleinberg (2010).

Multithreading. When possible, META algorithms and applications are parallelized using C++ threads to make full use of available resources.

[2] `https://meta-toolkit.org/nlp-demo.html`
[3] `http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html`

7.2 Usability

Consistency across components is a key feature that allows META to work well with large datasets. This is accomplished via a three-layer architecture. On the first layer, we have tokenizers, analyzers, and all the text processing that accompanies them. Once a document representation is determined, this tool chain is run on a corpus. The indexes are the second layer; they provide an efficient format for storing processed data. The third layer—the application layer—interfaces solely with indexes. This means that we may use the same index for running an SVM as we do to evaluate a ranking function, *without processing the data again*.

Since all applications use these indexes, META supports out-of-core classification with some classifiers. We ran our large classification dataset that doesn't fit in memory—Webspam (Webb et al., 2006)—using the `sgd` classifier. Where LIBLINEAR failed to run, META was able to finish the classification in a few minutes.

Besides using META's rich built-in feature generation, it is possible to directly use LIBSVM-formatted data. This allows preprocessed datasets to be run using any of META's algorithms. Additionally, META's `forward_index` (used for classification), is easily convertible to LIBSVM format. The reverse is also true: you may do feature generation with META, and use it to generate input for any other program that supports LIBSVM format.

META is hosted publicly on GitHub[4], which provides the project with community involvement through its bug/issue tracker and fork/pull request model. Its API is heavily documented[5], allowing the creation of Web-based applications (listed in section 7.1). The project website contains several tutorials that cover the major aspects of the toolkit[6] to enable users to get started as fast as possible with little friction. Additionally, a public forum[7] is accessible for all users to view and participate in user support topics, community-written documentation, and developer discussions.

A major design point in META is to allow for most of the functionality to be configured via a configuration file. This enables minimal effort exploratory data analysis without having to write (or recompile) any code. Designing the code in this way also encourages the components of the system to be pluggable: the entire indexing process, for example, consists of several modular layers which can be controlled by the configuration file.

An example snippet of a config file is given below; this creates a bigram part-of-speech analyzer. Multiple `[[analyzers]]` sections may be added, which META automatically combines while processing input.

[4] `https://github.com/meta-toolkit/meta/`
[5] `https://meta-toolkit.org/doxygen/namespaces.html`
[6] `https://meta-toolkit.org/`
[7] `https://forum.meta-toolkit.org/`

	CoreNLP			MeTA		
	Training	Testing	F_1	Training	Testing	F_1
Greedy	7m 27s 8.85 GB	18.6s 1.53 GB	86.7	17m 31s 0.79 GB	12.9s 0.29 GB	86.9
Beam (4)	6h 10m 43s 10.84 GB	46.8s 3.83 GB	89.9	2h 17m 25s 2.29 GB	59.2s 0.94 GB	88.1

Table 7.2: (NLP) Training/testing performance for the shift-reduce constituency parsers. All models were trained for 40 iterations on the standard training split of the Penn Treebank. Accuracy is reported as labeled F_1 from `evalb` on section 23.

```
[[analyzers]]
method = "ngram-pos"
ngram = 2
filter = [{type = "icu-tokenizer"},
          {type = "ptb-normalizer"}]
crf-prefix = "crf/model/folder"
```

A simple class hierarchy allows users to add filters, analyzers, ranking functions, and classifiers with full integration to the toolkit (e.g. one may specify user-defined classes in the config file). The process for adding these is detailed in the MeTA online tutorials.

This low barrier of entry experiment setup ease led to MeTA's use in text mining and analysis MOOCs reaching over 40,000 students[8,9].

Multi-language support is hard to do correctly. Many toolkits sidestep this issue by only supporting ASCII text or the OS language; MeTA supports multiple (non-romance) languages by default, using the industry standard ICU library[10]. This allows MeTA to tokenize arbitrarily-encoded text in many languages.

Unit tests ensure that contributors are confident that their modifications do not break the toolkit. Unit tests are automatically run after each commit and pull request, so developers immediately know if there is an issue (of course, unit tests may be run manually before committing). The unit tests are run in a continuous integration setup where MeTA is compiled and run on Linux, Mac OS X[11], and Windows[12] under a variety of compilers and software development configurations.

[8] `https://www.coursera.org/course/textretrieval`
[9] `https://www.coursera.org/course/textanalytics`
[10] `http://site.icu-project.org/`
[11] `https://travis-ci.org/meta-toolkit/meta`
[12] `https://ci.appveyor.com/project/skystrife/meta`

	Extra Data	Accuracy
Human annotators		97.0%
CoreNLP	✓	97.3%
LTag-Spinal		97.3%
SCCN	✓	97.5%
METa (CRF)		97.0%
METa (AP)		96.9%

Table 7.3: (NLP) Part-of-speech tagging token-level accuracies. "Extra data" implies the use of large amounts of extra unlabeled data (e.g. for distributional similarity features).

	Docs	Size	$\|D\\|_{avg}$	$\|V\|$
Blog06	3,215,171	26 GB	782.3	10,971,746
Gov2	25,205,179	147 GB	515.5	21,203,125

Table 7.4: (IR) The two TREC datasets used. Uncleaned versions of blog06 and gov2 were 89 GB and 426 GB respectively.

7.3 Experiments

We evaluate METa's performance in NLP, IR, and ML tasks. All experiments were performed on a workstation with an Intel(R) Core(TM) i7-5820K CPU, 16 GB of RAM, and a 4 TB 5900 RPM disk.

METa's part-of-speech taggers for English provide quite reasonable performance. It provides a linear-chain CRF tagger (CRF) as well as an averaged perceptron based greedy tagger (AP). We report the token level accuracy on sections 22–24 of the Penn Treebank, with a few prior model results trained on sections 0–18 in Table 7.3. "Human annotators" is an estimate based on a 3% error rate reported in the Penn Treebank README and is likely overly optimistic (Manning, 2011). CoreNLP's model is the result of Manning (2011), LTag-Spinal is from Shen et al. (2007), and SCCN is from Søgaard (2011). Both of METa's taggers are within 0.6% of the existing literature.

METa and CoreNLP both provide implementations of shift-reduce constituency parsers, following the framework of Zhu et al. (2013). These can be trained greedily or via beam search. We compared the parser implementations in METa and CoreNLP along two dimensions—speed, measured in wall time, and memory consumption, measured as maximum resident set size—for both training and testing a greedy and beam search parser (with a beam size of 4). Training was performed on the standard training split of sections 2–21 of the Penn Treebank, with section 22 used as a development set (only used by CoreNLP). Section 23 was held out for evaluation. The results are summarized in Table 7.2.

METa consistently uses less RAM than CoreNLP, both at training time and testing time. Its training time is slower than CoreNLP for the greedy parser, but less than half of CoreNLP's training time for the beam parser. METa's beam parser has worse labeled F_1 score, likely the result of its simpler model averaging

	Indri	Lucene	MeTA
Blog06	55m 40s	20m 23s	11m 23s
Gov2	8h 13m 43s	1h 59m 42s	1h 12m 10s

Table 7.5: (IR) Indexing speed.

	Indri	Lucene	MeTA
Blog06	31.02 GB	2.06 GB	2.84 GB
Gov2	170.50 GB	11.02 GB	10.24 GB

Table 7.6: (IR) Index size.

strategy[13]. Overall, however, MeTA's shift-reduce parser is competitive and particularly lightweight.

MeTA's IR performance is compared with two well-known search engine toolkits: Lucene's latest version 5.5.0[14] and Indri's version 5.9 (Strohman et al., 2005)[15].

We use the TREC blog06 (Ounis et al., 2006) permalink documents and TREC gov2 corpus (Clarke et al., 2004). To ensure a more uniform indexing environment, all HTML is cleaned before indexing. In addition, each corpus is converted into a single file with one document per line to reduce the effects of many file operations.

During indexing, terms are lower-cased, stop words are removed from a common list of 431 stop words, Porter2 (MeTA) or Porter (Indri, Lucene) stemming is performed, a maximum word length of 32 characters is set, original documents are not stored in the index, and term position information is not stored[16].

We compare the following: indexing speed (Table 7.5), index size (Table 7.6), query speed (Table 7.7), and query accuracy (Table 7.8) with BM25 using $k_1 = 0.9$ and $b = 0.4$. We use the standard TREC queries associated with each dataset and score each system's search results with the usual `trec_eval` program[17].

MeTA leads in indexing speed, though we note that MeTA's default indexer is multithreaded and Lucene does not provide a parallel one[18]. MeTA creates the smallest index for gov2 while Lucene creates the smallest index for blog06; Indri greatly lags behind both. MeTA follows Lucene closely in retrieval speed, with Indri again lagging. As expected, query performance between the three systems is relatively even, and we attribute any small difference in MAP or precision to idiosyncrasies during tokenization.

MeTA's ML performance is compared with liblinear (Fan et al., 2008), scikit-learn (Pedregosa et al., 2011), and SVMMulticlass[19]. We focus on linear classification with SVM across these tools (MAL-

[13]At training time, both CoreNLP and MeTAperform model averaging, but MeTA computes the average over all updates and CoreNLP performs cross-validation over a default of the best 8 models on the development set.

[14]`http://lucene.apache.org/`

[15]Indri 5.10 does not provide source code packages and thus could not be used. It is also known as Lemur.

[16]For Indri, we are unable to disable positions information storage.

[17]`http://trec.nist.gov/trec_eval/`

[18]Additionally, we did not feel that writing a correct and threadsafe indexer as a user is something to be reasonably expected.

[19]`http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html`

	Indri	Lucene	META
Blog06	55.0s	1.60s	3.67s
Gov2	24m 6.73s	57.53s	1m 3.98s

Table 7.7: (IR) Query speed.

	Indri		Lucene		META	
	MAP	P@10	MAP	P@10	MAP	P@10
Blog06	29.13	63.20	29.10	63.60	32.34	64.70
Gov2	25.96	53.69	30.23	59.26	29.97	57.43

Table 7.8: (IR) Query performance via Mean Average Precision and Precision at 10 documents.

LET (McCallum, 2002) does not provide an SVM, so it is excluded from the comparisons). Statistics for the four ML datasets can be found in Table 7.9.

The 20news dataset (Lang, 1995)[20] is split into its standard 60% training and 40% testing sets by post date. The Blog dataset (Schler et al., 2006) is split into 80% training and 20% testing randomly. Both of these two textual datasets were preprocessed using META using the same settings from the IR experiments.

The rcv1 dataset (Lewis et al., 2004) was processed into a training and testing set using the `prep_rcv1` tool provided with Leon Bottou's SGD tool[21]. The resulting training set has 781,265 documents and the testing set has 23,149. The Webspam corpus (Webb et al., 2006) consists of the subset of the Webb Spam Corpus used in the Pascal Large Scale Learning Challenge[22]. The corpus was processed using the provided `convert.py` into byte trigrams. The first 80% of the resulting file is used for training and the last 20% for testing.

In Table 7.10, we can see that META performs well both in terms of speed and accuracy. Both LIBLINEAR and SVMMULTICLASS were unable to produce models on the Webspam dataset due to memory limitations and lack of a minibatch framework. For SCIKIT-LEARN and META, we broke the training data into 4 equal sized batches and ran one iteration of SGD per batch. The timing result includes the time to load each chunk into memory; for META this is from its forward-index format[23] and for SCIKIT-LEARN this is from LIBSVM-formatted text files.

[20] `http://qwone.com/~jason/20Newsgroups/`
[21] `http://leon.bottou.org/projects/sgd`
[22] `ftp://largescale.ml.tu-berlin.de/largescale/`
[23] It took 12m 24s to generate the index.

	Docs	Size	k	Features
20news	18,846	86 MB	20	112,377
Blog	19,320	778 MB	3	548,812
rcv1	804,414	1.1 GB	2	47,152
Webspam	350,000	24 GB	2	16,609,143

Table 7.9: (ML) Datasets used for k-class categorization.

	liblinear	scikit	SVM$^{\text{mult}}$	MeTA
20news	79.4% 2.58s	74.3% 0.326s	67.1% 2.54s	80.1% 0.648s
Blog	75.8% 61.3s	76.2% 0.801s	72.2% 17.5s	72.2% 1.11s
rcv1	94.7% 17.6s	94.0% 1.66s	83.6% 2.01s	94.8% 3.44s
Webspam	✗	97.4% 11m 52s	✗	99.4% 1m 16s

Table 7.10: (ML) Accuracy and speed classification results. Reported time is to both train and test the model. For all except Webspam, this excludes IO.

7.4 Contributions from this Thesis

The three main contributions to this thesis are all contained in MeTA. This section simply shows how to use the configuration file to adjust settings for each work. For the most up-to-date information, please consult MeTA's home page, `https://meta-toolkit.org/`.

7.4.1 Structural Parse Tree Features

Structural parse tree features (i.e., skeleton and annotated skeleton features) are able to be used in MeTA. Additionally, we have a few other baseline parse tree features to choose from. Modify the `features` setting in the `analyzer` block. Note that it is possible to combine multiple tree features together in this section.

```
[[analyzers]]
method = "tree"
filter = [{type = "icu-tokenizer"}, {type = "ptb-normalizer"}]
features = ["skel", "subtree"]
tagger = "path/to/greedy-tagger/model"
parser = "path/to/sr-parser/model"
```

The possible values for `features` are:

- `branch`, branch featurizer: analyzes parse trees by the number of productions at each interior node.
- `depth`, depth featurizer: analyzes parse trees by their depth
- `skel`, skeleton featurizer: analyzes parse trees by their structural subtree patterns (Massung et al., 2013)
- `semi-skel`, semi-skeleton featurizer: analyzes parse trees by their annotated (keeping the root subtree node) structural patterns (Massung et al., 2013)
- `subtree`, subtree featurizer: analyzes parse trees using their rewrite rules.
- `tag`, tag featurizer: analyzes parse trees by counts of their interior node labels.

7.4.2 Syntactic Diff

SYNTACTICDIFF (Massung and Zhai, 2015) can directly be used with META to generate text features with respect to a reference corpus. Its configuration file setting is below:

```
[diff]
n-value = 3
max-edits = 3
base-penalty = 0.0 # base penalty is for any edit
insert-penalty = 0.0
substitute-penalty = 0.0
remove-penalty = 0.0
```

By default, all the penalties are zero, but they can be arbitrarily set in this configuration. Additionally, the following language model section is needed to represent the reference corpus:

```
[language-model]
arpa-file = "../data/english-sentences.arpa"
binary-file-prefix = "english-sentences-"
```

The `arpa-file` is a prelearned language model that can be from any other corpus. This is what is used to determine where to make edits to each sentence inspected by SYNTACTICDIFF.

7.4.3 Cross-Context Lexical Analysis

CCLA (Massung et al., 2017) is also available under a configuration setting.

```
[ccla]
corpora = ["imdb", "yelp"]
context-labels = ["positive", "negative"]
k = 500
scoring-function = "nearest-neighbors" # or "ndcg"
annotation = "embedding"
```

The `corpora` parameter is a list of META datasets; `context-labels` is a metadata field value that is used to split documents into contexts. The k parameter is the top-k nearest neighbors or NDCG@k parameter for the scoring function. Currently, only the embedding annotation is supported.

7.5 Conclusions and Future Work

META is a valuable resource for text mining applications; it is a viable and competitive alternative to existing toolkits that unifies algorithms from natural language processing, information retrieval, and machine learning. META is an extensible, consistent framework that enables quick development of complex application systems. It has demonstrated its usefulness in courses from twenty students to MOOCs with tens of thousands of students; it can efficiently operate on terabytes of data while also supporting novice users through configuration file manipulation.

All of the work in this thesis is contained in META. It will always remain a free and open-source toolkit for text retrieval and analysis.

CHAPTER 8

CONCLUSIONS AND OPEN QUESTIONS

The importance and prevalence of text data has been greatly emphasized in this thesis and the papers that it contains. Converting raw strings to quantized feature vectors or sequences is a crucial step in any text processing application. Example applications we explored in this thesis are native language identification, grammatical error correction, summarization via topic analysis, authorship attribution, essay scoring, sentiment analysis, analysis of semantic change, comparative lexical analysis over context, and word embedding comparison.

In chapter 2, we gave a very broad, general introduction and literature survey to text representation and techniques. In contrast to chapter 2, chapter 3 explored a particular subfield of text mining that benefits from advances in text representation: non-native text mining. This subfield operates on text data that has been produced by writers in a non-native language. Many of the non-topical features explored for non-native text analysis performed well due to their ability to capture the concept of "nativeness" or "fluency", which inherently makes them interpretable.

As mentioned in chapter 1, a major limitation in existing text representation methods is the dependence on topic-based features. When we diverge from topic-based features, we often lose interpretability or explanatory power. To address this issue, non-topical text representations and mining methods were introduced and studied in chapters 4, 5, and 6.

We began with structural parse tree features in chapter 4, comparing combinations of simple n-gram text representation models with new and existing tree features. We showed that the novel structural tree features are most effective and when combined with a simpler lexical model, capturing multiple perspectives of the same text. Using these new methods, we displayed performance gains on existing corpora across domains. This demonstrated the generality and usefulness of our features. We showed that the new structural features combine better with simple features than existing tree representations such as rewrite rules. The interpretability of the skeleton features is a key point; we showed common phrases for each L1 that would not be captured by existing methods, yet were easily understandable by humans.

SYNTACTICDIFF was introduced in chapter 5. It is a novel, efficient, and general framework for many

text mining tasks that examines syntactic differences between target text and a reference background collection. These differences are captured in weighted edit operations. These text edits can not only be used to generate an alternative representation of text data that is complementary with the content-based representation, but also support a wide range of interesting novel applications. We evaluated the generality and effectiveness of SYNTACTICDIFF using three distinct tasks: grammatical error correction, corpus summarization, and classification. In all areas, SYNTACTICDIFF provided concrete advantages, clearly demonstrating its empirical benefit. Importantly, the features produced by SYNTACTICDIFF are human-interpretable; inspecting top features for a given label provide immediate insight into why they were chosen and what makes a particular class label different than other labels.

Cross-Context Comparative Analysis (CCLA) was discussed in chapter 6. CCLA generally refers to any analysis of term meaning or term representation in different contexts, especially for understanding the differences and similarities in *multiple* contexts. We formalized this notion with a general framework that accommodates any notion of context, any similarity function, and any type of word annotation. This enables many new applications all under the same framework (e.g. development of a common toolkit to support all applications), including analysis of semantic change, comparative analysis of meaning over context, and word embedding evaluation. CCLA is a particularly useful tool when we wish to examine (e.g.) word annotations that are not inherently topic-based. For example, we derived the understandable notion of word embedding *stability* using the CCLA framework and showed that it is correlated with word similarity and analogy performance.

In chapter 7, we overviewed META, a unified toolkit for text retrieval and analysis. It is a viable and competitive alternative to existing toolkits that unifies algorithms from natural language processing, information retrieval, and machine learning. META is an extensible, consistent framework that enables quick development of complex application systems. It has demonstrated its usefulness in courses from twenty students to MOOCs with tens of thousands of students; it can efficiently operate on terabytes of data while also supporting novice users through configuration file manipulation. All of the work in this thesis is contained in META. It will always remain a free and open-source toolkit for text retrieval and analysis.

Although many advances in explanatory text mining and representation are included as part of this thesis, there are still many open questions and avenues for future work.

We would like to explore structural features in tree structures other than PCFGs (such as dependency parses or XML documents), as well in other domains such as clustering and information retrieval. Additionally, we would be interested in seeing how the features respond to dimensionality reduction techniques, and if this can further increase the interpretability of the top features per class.

For SYNTACTICDIFF, there are many interesting future directions to further explore, particularly in

leveraging such a new representation in many other applications, exploring different configurations for comparative text analysis, and further generalizing the framework to capture more semantic meaning. It should be straightforward to improve the candidate generation efficiency via beam search. Lastly, exploring the possibility of learning new penalty types during training is desirable. To approach this, we can break the definition of a penalty into context and an argument. One context could be surrounding part of speech tags, and the argument is the current word examined in an edit operation. Once SYNTACTICDIFF operates in this format, we can arbitrarily create penalties.

CCLA opens up interesting new directions for further study, especially in additional applications. One use is to investigate framing bias on political viewpoints. Another is a more fine-grained comparative analysis over specific products as opposed to movies or businesses. Term scoring can be further taken advantage of in sentiment valence prediction. Pablos et al. (2016) use word vector similarity to create sentiment valence scores per term, but they only consider similarity with a manually-chosen positive and negative word. Word sense disambiguation is another unvisited technique, and CCLA's notion of context could help determine which words have multiple senses. Using CCLA as a tool in a larger system is desirable, such as learning to automatically partition a corpus to maximize word differences, or using it for event detection when tones shift from a monitored stream. We want to investigate embedding comparisons further using larger training data and automatically determine an optimal dimensionality or window size given new scoring functions.

Finally, a detailed analysis of combining multiple interpretable features would be hugely beneficial to downstream tasks. Determining programmatically *why* features are interpretable would greatly help in their design and effectiveness.

REFERENCES

Emil Abrahamsson, Timothy Forni, Maria Skeppstedt, and Maria Kvist. Medical Text Simplification using Synonym Replacement: Adapting Assessment of Word Difficulty to a Compounding Language. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 57–65, Gothenburg, Sweden, April 2014.

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment Analysis of Twitter Data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-96-1.

Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

Roee Aharoni, Moshe Koppel, and Yoav Goldberg. Automatic Detection of Machine Translated Text and Translation Quality Estimation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 289–295, Baltimore, Maryland, June 2014.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On Smoothing and Inference for Topic Models. In *UAI*, 2009.

Yigal Attali and Jill Burstein. Automated Essay Scoring with e-rater v.2.0. *Journal of Technology, Learning, and Assessment*, 4(3), 2006.

R Harald Baayen, H van Halteren, A Neijt, and F Tweedie. Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution. *Literary and Linguistic Computing*, 11(3):121–132, 1996.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to Explain Individual Classification Decisions. *Journal of Machine Learning Research*, 11: 1803–1831, August 2010.

Aayush Bansal, Ali Farhadi, and Devi Parikh. Towards Transparent Systems: Semantic Characterization of Failure Modes. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI*, pages 366–381, 2014.

Regina Barzilay and Lillian Lee. Bootstrapping Lexical Choice via Multiple-Sequence Alignment. In *Proceedings of EMNLP 2002*, pages 164–171, 2002.

Regina Barzilay and Lillian Lee. Learning to Paraphrase: An Unsupervised Approach Using Multiple-sequence Alignment. In *Proceedings of NAACL 2003*, pages 16–23, 2003.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013.

Ezra Black, Stephen Eubank, Hideki Kashioka, David M. Magerman, Roger Garside, and Geoffrey Leech. Beyond Skeleton Parsing: Producing a Comprehensive Large-Scale General-English Treebank With Full Grammatical Analysis. In *COLING*, pages 107–112, 1996.

D. Blanchard, J. Tetreault, D. Higgins, A. Cahill, and M Chodorow. TOEF11: a Corpus of Non-Native English. Technical report, Educational Testing Service, 2013. `https://www.ets.org/`.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

Phil Blunsom and Trevor Cohn. Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1204–1213, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Julian Brooke and Graeme Hirst. Robust, Lexicalized Native Language Identification. In *COLING*, pages 391–408, 2012.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based N-gram Models of Natural Language. *Comput. Linguist.*, 18(4):467–479, 1992.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. Distributional Semantics in Technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 136–145. Association for Computational Linguistics, 2012.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 507–517, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Rich Caruana and Alexandru Niculescu-Mizil. Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria. In *ROCAI'04*, pages 9–18, 2004.

Changyou Chen, Wray Buntine, Nan Ding, Lexing Xie, and Lan Du. Differential Topic Models. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):230–242, 2015.

Miao Chen and Klaus Zechner. Computing and Evaluating Syntactic Complexity Features for Automated Scoring of Spontaneous Non-Native Speech. In *ACL*, pages 722–731, 2011.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel Tetreault. Problems in Evaluating Grammatical Error Detection Systems. In *Proceedings of COLING 2012*, pages 611–628, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2004 Terabyte Track. In *TREC*, 2004.

Jacob Cohen. Weighted Kappa: Nominal Scale Agreement Provision for Scaled Disagreement or Partial Credit. *Psychological Bulletin*, 70:213–220, 1968.

Michael Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP*, 2002.

Michael Collins and Nigel Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels Over Discrete Structures, and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011. ISSN 1532-4435.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. A Machine Learning Approach to the Automatic Evaluation of Machine Translation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 148–155, Toulouse, France, July 2001. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. Grammatical Error Correction with Alternating Structure Optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 915–923, Stroudsburg, PA, USA, 2011a. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. Correcting Semantic Collocation Errors with L1-induced Paraphrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Edinburgh, Scotland, UK., July 2011b. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, 2013.

Mark Davies. The Corpus of Contemporary American English as the First Reliable Monitor Corpus of English. *Literary and Linguistic Computing*, 25(4):447, 2010.

Felice Dell'Orletta, Martijn Wieling, Giulia Venturi, Andrea Cimino, and Simonetta Montemagni. Assessing the Readability of Sentences: Which Corpora and Features? In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–173, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Semire Dikli. An Overview of Automated Scoring of Essays. *Journal of Technology, Learning, and Assessment*, 5(1), 2006.

David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008. ISSN 1532-4435.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 30–35. Association for Computational Linguistics, August 2016.

Song Feng, Ritwik Banerjee, and Yejin Choi. Syntactic Stylometry for Deception Detection. In *ACL*, pages 171–175, 2012.

J. Foulds, L. Boyles, C. DuBois, P. Smyth, and M. Welling. Stochastic Collapsed Variational Bayesian Inference for Latent Dirichlet Allocation. In *KDD*, 2013.

The Hewlett Foundation. ASAP: Automated Student Essay Prize, 5 2012. URL `http://www.kaggle.com/c/asap-aes`.

Michael Gamon. Using Mostly Native Data to Correct Errors in Learners' Writing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171, Los Angeles, California, June 2010. Association for Computational Linguistics.

Michael Gamon, Anthony Aue, and Martine Smets. Sentence-level MT Evaluation Without Reference Translations: Beyond Language Modeling. In *Proceedings of the Tenth European Association for Machine Translation*, pages 103–111, Budapest, Hungary, 2005a.

Michael Gamon, Anthony Aue, and Martine Smets. Sentence-level MT Evaluation Without Reference Translations: Beyond Language Modeling. In *In European Association for Machine Translation (EAMT)*, 2005b.

T. L. Griffiths and M. Steyvers. Finding Scientific Topics. *PNAS*, 101, April 2004.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1489–1501, Berlin, Germany, August 2016.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns Without Candidate Generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4.

Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123814790, 9780123814791.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. Predicting Grammaticality on an Ordinal Scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 174–180, Baltimore, Maryland, 2014.

Andrea Horbach, Jonathan Poitz, and Alexis Palmer. Using Shallow Syntactic Features to Measure Influences of L1 and Proficiency Level in EFL Writings. In *Proceedings of 4th NLP4CALL Workshop (NLP for Computer-Aided Language Learning)*, pages 21–34, Vilnius, Lithuania, 2015.

Hwee Tou Ng and Siew Mei Wu and Yuanbin Wu and Christian Hadiwinoto and Joel Tetreault. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2013*, 2013.

Shin Ishikawa. Vocabulary in Interlanguage: A study on Corpus of English Essays Written by Asian University Students (CEEAUS). In *Phraseology, Corpus Linguistics and Lexicography*, Phraseology, pages 87–100, 2009.

Shinichiro Ishikawa. The ICNALE and Sophisticated Contrastive Interlanguage Analysis of Asian Learners of English. In *Learner Corpus Studies in Asia and the World*, pages 91–118, 2013.

Jing Jiang and ChengXiang Zhai. A Systematic Exploration of the Feature Space for Relation Extraction. In *In Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 113–120, 2007.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models. In *NIPS*, pages 641–648, 2006.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to NLP, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, 2000.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 941–951, Berlin, Germany, August 2016. Association for Computational Linguistics.

Sangkyum Kim, Hyungsul Kim, Tim Weninger, Jiawei Han, and Hyun Duk Kim. Authorship Classification: a Discriminative Syntactic Tree Mining Approach. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 455–464, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal Analysis of Language through Neural Language Models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65. Association for Computational Linguistics, June 2014.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302. 2015.

Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. Determining an Author's Native Language by Mining a Text for Errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 624–628. ACM, 2005.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *Journal of the American Society of Information Science and Technology.*, 60(1):9–26, January 2009.

Taku Kudo and Yuji Matsumoto. A Boosting Algorithm for Classification of Semi-Structured Text. In *EMNLP*, pages 301–308, 2004.

Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R. Klemmer. Peer and Self Assessment in Massive Online Classes. *ACM Trans. Comput.-Hum. Interact.*, 20(6):33:1–33:31, 2013. ISSN 1073-0516.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically Significant Detection of Linguistic Change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 625–635, 2015.

Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Freshman or Fresher? Quantifying the Geographic Variation of Language in Online Social Media. In *Proceedings of the Tenth International Conference on Web and Social Media*, pages 615–618, 2016.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Ken Lang. Newsweeder: Learning to Filter Netnews. In *ICML*, 1995.

Theodoros Lappas and Michail Vlachos. Customizing Search Results for Non-native Speakers. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1829–1833, Maui, Hawaii, 2012.

Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, Beijing, China*, pages 1188–1196, 2014.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel R. Tetreault. *Automated Grammatical Error Detection for Language Learners, Second Edition*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2014.

John Lee and Stephanie Seneff. Automatic Grammar Correction for Second-Language Learners. In *In Interspeech ISCA*, 2006.

John Lee, Ming Zhou, and Xiaohua Liu. Detection of Non-Native Sentences Using Machine-Translated Training Data. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 93–96, Rochester, New York, 2007.

Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. ISSN 2307-387X.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR*, 5, December 2004.

Heng Li and Nils Homer. A Survey of Sequence Alignment Algorithms for Next-generation Sequencing. *Briefings in Bioinformatics*, 11(5):473–483, 2010.

Adam Lopez. Statistical Machine Translation. *ACM Comput. Surv.*, 40(3):8:1–8:49, 2008.

Thang Luong, Richard Socher, and Christopher Manning. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113. Association for Computational Linguistics, 2013.

Collin F. Lynch, Kevin D. Ashley, and Min Chi. *Can Diagrams Predict Essay Grades?*, pages 260–265. Springer International Publishing, Cham, 2014. ISBN 978-3-319-07221-0.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011a. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, June 2011b.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Exploring Grammatical Error Correction with Not-So-Crummy Machine Translation. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 44–53, Montréal, Canada, June 2012. Association for Computational Linguistics.

Christopher D. Manning. Part-of-speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Proc. CICLing*, 2011.

Sean Massung and ChengXiang Zhai. SyntacticDiff: Operator-Based Transformation for Comparative Text Mining. In *Proceedings of the 3rd IEEE International Conference on Big Data*, pages 571–580, 2015.

Sean Massung and ChengXiang Zhai. Non-Native Text Analysis: A Survey. *The Journal of Natural Language Engineering*, 22(2):163–186, 2016.

Sean Massung, ChengXiang Zhai, and Julia Hockenmaier. Structural Parse Tree Features for Text Representation. In *IEEE 7th International Conference on Semantic Computing*, pages 9–13, Irvine, CA, 2013.

Sean Massung, Chase Geigle, and ChengXiang Zhai. MeTA: A Unified Toolkit for Text Retrieval and Analysis. In *Proceedings of ACL-2016 System Demonstrations*, pages 91–96. Association for Computational Linguistics, 8 2016.

Sean Massung, Chase Geigle, and ChengXiang Zhai. Cross-Context Lexical Analysis. In *Submission to EMNLP*. Association for Computational Linguistics, 2017.

Suzanne Maxwell. The Effects of Two Types of Text Modification on English Language Learners' Reading Comprehension: Simplification versus Elaboration. Master's thesis, Hamline University, Saint Paul, Minnesota, 2011.

Andrew Kachites McCallum. MALLET: A Machine Learning for Language Toolkit, 2002. `http://mallet.cs.umass.edu/`.

Qiaozhu Mei and ChengXiang Zhai. A Mixture Model for Contextual Text Mining. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 649–655, 2006.

Qiaozhu Mei, Chao Liu, Hang Su, and ChengXiang Zhai. A Probabilistic Approach to Spatiotemporal Theme Pattern Mining on Weblogs. In *Proceedings of the 15th international conference on World Wide Web*, pages 533–542. ACM, 2006.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR).*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013b.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, 2013c.

David Mimno and Andrew McCallum. Topic Models Conditioned on Arbitrary Features with Dirichlet-multinomial Regression. In *Proceedings of UAI (2008)*, 2008.

Alessandro Moschitti. Making Tree Kernels Practical for Natural Language Learning. In *EACL*, 2006.

Ryo Nagata and Keisuke Sakaguchi. Phrase Structure Annotation and Parsing for Learner English. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1837–1847, Berlin, Germany, August 2016. Association for Computational Linguistics.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed Algorithms for Topic Models. *JMLR*, 10, December 2009.

Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization. *SIGIR Forum*, 31(SI):67–73, July 1997. ISSN 0163-5840.

Peter Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann Publishers Inc., 1st edition, 1992.

I. Ounis, C. Macdonald, M. de Rijke, G. Mishne, and I. Soboroff. Overview of the TREC 2006 Blog Track. In *TREC*, 2006.

Aitor García Pablos, Montse Cuadros, and German Rigau. A Comparison of Domain-based Word Polarity Estimation using different Word Embeddings. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*, 2016.

Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

Kristen Parton, Joel Tetreault, Nitin Madnani, and Martin Chodorow. E-rating Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 108–115, Edinburgh, Scotland, 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *JMLR*, 12, 2011.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1001-9.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

Sarah E. Petersen and Mari Ostendorf. Text Simplification for Language Learners: A Corpus Analysis. In *Proceedings of the Workshop on Speech and Language Technology for Education*, pages 69–72, Pittsburgh, Pennsylvania, 2007.

Martin Porter. The English Porter2 Stemming Algorithm, 10 2012. URL `http://snowball.tartarus.org/algorithms/english/stemmer.html`.

Donald Powers, Jill Burstein, Martin Chodorow, Mary Fowles, and Karen Kukich. Stumping e-rater: Challenging the Validity of Automated Essay Scoring. Technical Report GRE No. 98-08bP, ETS RR-01-03, Educational Testing Service, Princeton, New Jersey, 2001.

Ella Rabinovich, Sergiu Nisioi, Noam Ordan, and Shuly Wintner. On the Similarities Between Native, Non-native and Translated Texts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1881, Berlin, Germany, August 2016. Association for Computational Linguistics.

Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. Authorship Attribution Using Probabilistic Context-Free Grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 38–42, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *TREC*, 1994.

Mark Robson. The English Effect, April 2014. https://www.britishcouncil.org/sites/default/files/english-effect-report-v2.pdf.

Stéphane Ross, Paul Mineiro, and John Langford. Normalized Online Learning. In *UAI*, 2013.

Alla Rozovskaya and Dan Roth. Training Paradigms for Correcting Errors in Grammar and Usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 154–162, Los Angeles, California, June 2010. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. Joint Learning and Inference for Grammatical Error Correction. In *EMNLP*, pages 791–802, 2013.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. Effects of Age and Gender on Blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, 2006.

Nihar Shah, Joseph Bradley, Sivaraman Balakrishnan, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. Some Scaling Laws for MOOC Assessments. In *KDD Workshop on Data Mining for Educational Assessment and Feedback*, 2014.

Libin Shen, Giorgio Satta, and Aravind Joshi. Guided Learning for Bidirectional Sequence Classification. In *ACL*, June 2007.

Advaith Siddharthan. A Survey of Research on Text Simplification. *International Journal of Applied Linguistics*, pages 259–298, 2014.

Anders Søgaard. Semi-supervised Condensed Nearest Neighbor for Part-of-Speech Tagging. In *ACL-HLT*, June 2011.

Efstathios Stamatatos. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009.

Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. Indri: A Language-Model Based Search Engine for Complex Queries (Extended Version). IR 407, University of Massachusetts, 2005.

Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, and Chin-Yew Lin. Detecting Erroneous Sentences using Automatically Mined Sequential Patterns. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic, 2007.

Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields. In *Foundations and Trends in Machine Learning*, 2012.

Ben Swanson and Eugene Charniak. Native Language Detection with Tree Substitution Grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 193–197, Stroudsburg, PA, USA, 2012.

Luchen Tan, Haotian Zhang, Charles L. A. Clarke, and Mark D. Smucker. Lexical Comparison Between Wikipedia and Twitter Corpora by Using Word Embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*, pages 657–661, 2015.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. A Report on the First NLI Shared Task. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, 2013.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Oren Tsur and Ari Rappoport. Using Classifier Features for Studying the Effect of Native Language on the Choice of Written Second Language Words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, CACLA '07, pages 9–16, Stroudsburg, PA, USA, 2007.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty. In *ACLIJCNLP*, 8 2009.

Rui Wang and Günter Neumann. Recognizing Textual Entailment using Sentence Similarity based on Dependency Tree Skeletons. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 36–41, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Sida Wang and Christopher D. Manning. Baselines and bigrams: simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Steve Webb, James Caverlee, and Carlton Pu. Introducing the Webb Spam Corpus: Using Email Spam to Identify Web Spam Automatically. In *CEAS*, July 2006.

Randy West, Y. Albert Park, and Roger Levy. Bilingual Random Walk Models for Automated Grammar Correction of ESL Author-produced Text. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, pages 170–179, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 9781937284039.

Sze-Meng Jojo Wong and Mark Dras. Contrastive Analysis and Native Language Identification. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 53–61, 2009.

Sze-Meng Jojo Wong and Mark Dras. Exploiting Parse Structures for Native Language Identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1600–1610, Stroudsburg, PA, USA, 2011.

Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. Exploring Adaptor Grammars for Native Language Identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 699–709, Stroudsburg, PA, USA, 2012.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. Sentence Simplification by Monolingual Machine Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 1015–1024, Jeju Island, Korea, 2012.

Su-Youn Yoon and Suma Bhat. Assessment of ESL Learners' Syntactic Competence Based on Similarity Measures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 600–608, Jeju Island, Korea, July 2012.

Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Who, Where, When and What: Discover Spatio-temporal Topics for Twitter Users. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 605–613, 2013. ISBN 978-1-4503-2174-7.

ChengXiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.*, 22(2), April 2004.

ChengXiang Zhai and Sean Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA, 2016. ISBN 978-1-97000-117-4.

ChengXiang Zhai, Atulya Velivelli, and Bei Yu. A Cross-collection Mixture Model for Comparative Text Mining. In *Proceedings of ACM SIGKDD 2004*, pages 743–748, 2004. ISBN 1-58113-888-1.

Sendong Zhao, Quan Wang, Sean Massung, Ting Liu, and ChengXiang Zhai. Constructing and Embedding Abstract Event Causality Networks from Text Snippets. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*, February 2017.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature Selection for Text Categorization on Imbalanced Data. *SIGKDD Explor. Newsl.*, 6(1):80–89, June 2004. ISSN 1931-0145.

Guodong Zhou, Longhua Qian, and Jianxi Fan. Tree Kernel-based Semantic Relation Extraction with Rich Syntactic and Semantic Information. *Information Science*, 180(8):1313–1325, April 2010.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and Accurate Shift-Reduce Constituent Parsing. In *ACL*, 2013.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1353–1361, Beijing, China, 2010.