2017

Representations of Idioms for Natural Language Processing: Idiom type and token identification, Language Modelling and Neural Machine Translation

Giancarlo Salton
Technological University Dublin, giancarlo.salton@mydit.ie

Representations of Idioms for Natural Language Processing: Idiom type and token identification, Language Modelling and Neural Machine Translation

by

Giancarlo D. Salton

Supervisors: Dr. John D. Kelleher
Dr. Robert J. Ross

School of Computing
Dublin Institute of Technology

Thesis submitted for the degree of

Doctor of Philosophy

October 2017

To my father Adir J. Salton (*in memoriam*)

Declaration

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The work reported on in this thesis conforms to the principles and requirements of the DIT's guidelines for ethics in research.

DIT has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signature: _____ Date: _____

Acknowledgements

I would like to thank my family for their support and understanding: my mother Maria Rosa for always facing all the difficulties and challenges life has presented to her and my family with courage and giving us the strength to overcome them; my brother Guilherme Pedro whose happiness and persistence are always an inspiration; and to my auntie/godmother/second mother Dione for being always there when our parents could not be and for her great impact on making me the person I am today.

I would also like to thank my wife's family: Antônio, Mary, Amanda, Edgar and Gregory. Their support has made our life much easier during this journey.

I would also like to thank both my supervisors for their guidance, support and friendship: my lead supervisor, Prof. John D. Kelleher, the person who took the challenge to accept me as his student without knowing me in person until the day arrived in Dublin and who who pushed me to the very best of what I could achieve. And my second supervisor, Dr. Robert J. Ross, who has also been there since the very beginning and whose advises on both the scientific and practical side helped to make the journey much less frightening.

I need also to thank all the colleagues and friends that joined or left the group along the way and made our Friday evenings in the reading group so very enjoyable: Paddy, Niels, Colm, Eoin, Ivan, Pierre, Hao, Fei, Elizabeth, Lucas, Alex, Kristina, Caroline, Jack, Marek, Camille, Hector, Patrick, Senja, Irene and Guanhong. I also would like to thank the friends I made at the Focas Institute, especially those who also played in the tag-rugby team.

I would also like to thank Dr. Braulio A. de Melo who introduced me to research and has been a good friend ever since.

I also need to thank Vincent and Bernie Fitzgerald for their kindness, support and friendship since I arrived in Dublin.

On the business side, I would like to thank CAPES ("Coordenação de Aperfeiçoamento de Pessoal de Nível Superior") for my scholarship (Science Without Borders, proc n. 9050-13-2).

I also thank the staff at the School of Computing who helped me a lot during the journey and thought me much during the process.

Finally I left the biggest acknowledgement to my lovely wife Acassia. Without her I would not been able to get through this journey. She shared all the good and the bad, the best and the worst of this voyage. She smiled and cried with me but she never let me get my head down. Her love and support were fundamental for me to finish this work.

Dublin, September 2017.

Abstract

An idiom is a multiword expression (MWE) whose meaning is non-compositional, *i.e.*, the meaning of the expression is different from the meaning of its individual components. Idioms are complex constructions of language used creatively across almost all text genres. Idioms pose problems to natural language processing (NLP) systems due to their non-compositional nature, and the correct processing of idioms can improve a wide range of NLP systems.

Current approaches to idiom processing vary in terms of the amount of discourse history required to extract the features necessary to build representations for the expressions. These features are, in general, statistics extracted from the text and often fail to capture all the nuances involved in idiom usage.

We argue in this thesis that a more flexible representations must be used to process idioms in a range of idiom related tasks. We demonstrate that high-dimensional representations allow idiom classifiers to better model the interactions between global and local features and thereby improve the performance of these systems with regard to processing idioms.

In support of this thesis we demonstrate that distributed representa-

tions of sentences, such as those generated by a Recurrent Neural Network (RNN) greatly reduce the amount of discourse history required to process idioms and that by using those representations a "general" classifier, that can take any expression as input and classify it as either an idiomatic or literal usage, is feasible.

We also propose and evaluate a novel technique to add an attention module to a language model in order to bring forward past information in a RNN-based Language Model (RNN-LM). The results of our evaluation experiments demonstrate that this attention module increases the performance of such models in terms of the perplexity achieved when processing idioms. Our analysis also shows that it improves the performance of RNN-LMs on literal language and, at the same time, helps to bridge long-distance dependencies and reduce the number of parameters required in RNN-LMs to achieve state-of-the-art performance.

We investigate the adaptation of this novel RNN-LM to Neural Machine Translation (NMT) systems and we show that, despite the mixed results, it improves the translation of idioms into languages that require distant reordering such as German. We also show that these models are suited to small corpora for *in-domain* translations for language pairs such as English/Brazilian-Portuguese.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multiword expressions (MWEs) are expressions that carry a certain degree of idiosyncrasy in their properties and are generally acknowledged to cause problems to many areas of Natural Language Processing (NLP) (Sag et al., 2002). The idiosyncratic properties, or *idiomacity*, of MWEs refers to the deviation from the basic properties of the component words and, according to Baldwin and Kim (2010), five sub-types of idiomacity are recognized:

1. **Lexical**: This sub-type of idiomacity occurs when one or more word components do not belong to the lexicon that the MWE belongs to. For example, *ad hoc* is lexically idiomatic in English as none of the component words belong to the English lexicon.

2. **Syntactic**: Syntactic idiomacity occurs when the syntax of the expression cannot be directly derived from its components. For example, *by and large* is composed of a preposition (*by*) and an ad-

jective (*large*).

3. **Semantic**: Also referred to as *Idioms*, these MWE cannot have their meaning derived from their individual constituents. For example, *make a scene* has the figurative meaning of "to make a public display or disturbance" which is different from the meaning of both *make* and *scene*.

4. **Pragmatic**: Occurs when the MWE is associated with a particular context. For example, *good morning* is a greeting associated with mornings.

5. **Statistical**: Statistical idiomacity, also referred to as *conventionalisation* or *collocations*, occurs when the combination of words that form the MWE occurs with higher frequency than alternative phrasings of the same expression. For example, in English it is arbitrary to say *salt and pepper* in preference to *pepper and salt*.

Although the idiomacity sub-types are clearly distinct, it is important to note that most MWEs fall into a continuum of idiomacity and can be classified into more than one sub-type, as, for example, *ad hoc* which has both lexical and semantic idiomacity.

In this thesis we are interested in the semantic subtype, commonly referred to as Idioms. Idioms are often defined as a class of MWEs whose meaning is non-compositional although, in fact, most idioms also

present varying degrees of syntactic and statistical idiomacity. Therefore, there is no general consensus on the definition of the characteristics that covers all members of this class (Nunberg et al., 1994). In addition, a great number of idioms follow similar patterns and can be grouped into natural classes, while others cannot be easily associated with any group (Villavicencio et al., 2004). For example, idioms present different degrees of statistical idiomacity (or fixedness). Some idioms such as *by and large* ("on the whole; everything considered") *make a scene* and *kingdom come* ("into the next world, or the eternity"), are fixed to a degree that they are considered *words-with-spaces* (Sag et al., 2002), and therefore can be considered high-fixed. Conversely, there are idioms such as *hold fire* ("wait; calm down"), that accept different degrees of variation (Copestake et al., 2002) (*e.g.*, different lexical and syntactic choices) and thus are considered low-fixed. Although it is not the norm for idioms to have such variants (Fazly et al., 2009), variations in low-fixed idioms are too common to be ignored (Riehemann, 2001) by NLP systems. A complicating factor about idioms is that their syntactic and lexical variations are context dependent and most of the time unpredictable. Therefore, depending on the context, they often do not retain their figurative interpretation when they undergo such variations (Fazly and Stevenson, 2006).

Another remarkable property of idioms is that some expressions can also have a literal usage. For example, compare the two sentences below,

extracted from the VNC-Tokens dataset (Cook et al., 2008):

1. *No doubt you will have a word with me about this if you need further information*.

2. *The species name has two words:* Tyrannosaurus rex *is a familiar example*.

The first sentence carries an idiomatic usage for the expression *have a word* while the second sentence can only be interpreted literally. In general, idiomatic usages of an expression are more frequent than literal usages but, depending on the context where the expression is inserted, the literal usage might become much more frequent (Li and Sporleder, 2010a). For example, in a sports context, the expression *drop the ball* is expected to occur more times with its literal meaning than the figurative usage of "missing an opportunity". Additionally, some idioms might have two or more unrelated figurative meanings when the context is not considered (*e.g.*, when found listed in a dictionary) (Hashimoto and Kawahara, 2008). Examples of this phenomenon are *give up the ghost* (1- *die*; 2- *stop working*; 3- *give up the hope*) and *hang one on* (1- *punch someone hard*; 2- *get drunk*).

Given the idiosyncratic properties of idioms and their pervasiveness in human languages, broad-coverage NLP systems must explicitly process idioms in order to correctly interpret these expressions and produce appropriate naturalistic language (Villavicencio et al., 2005). To achieve

this, there are two main identification tasks involving idioms:

i) *idiom type* identification, which is the identification of expressions that can potentially be used as idioms, independent of the context they are inserted in;

ii) and *idiom token* identification, the disambiguation between idiomatic and literal usages of an expression in its context.

We are interested in idioms for the following reasons: (a) idioms are pervasive across different languages and text genres, enabling compact representations of large fragments of meaning (Salton et al., 2014b); (b) as a class, idioms are relatively frequent in daily use (Sporleder et al., 2010), although occurrences of individual expressions are relatively rare (examples of such expressions are *drop a line* - "to contact someone with a short email, message, or phone call" - and *find one's feet* - "to become used to a new situation or experience"); and (c) processing idioms correctly is still considered a major challenge in NLP, negatively affecting applications such as parsing, language understanding and machine translation, among others (Sag et al., 2002; Shutova et al., 2010; Muzny and Zettlemoyer, 2013; Salton et al., 2014a, 2016; Nerima et al., 2017).

Current approaches to processing idioms vary in terms of the complexity of representations and the length of the discourse history taken into consideration. For example, previous work in *idiom token* identification have shown that paragraphs surrounding the candidate expression

are an important feature in distinguishing between idiomatic and literal usages (*e.g.*, Li and Sporleder (2010a) and Peng et al. (2014)). In general, idioms are considered outliers to the discourse in which they are inserted as the non-compositionality of these expressions introduce words into the text that one would not expect to see in that discourse context. However, as pointed out by (Li and Sporleder, 2010a), in some discourse contexts the literal usage may become more frequent (although this does not exclude the possibility of using these expressions in an idiomatic sense) and simple representations (such as topics) will fail to capture all the nuances of an idiom. In addition, NLP systems often do not have access to such a large discourse history (*e.g.* a number of surrounding paragraphs) to extract representations of idioms.

Processing idioms (be it for *idiom type* identification, *idiom token* identification, language modelling or in machine translation) is difficult because idioms are complex structures that draw both on the local structure of the idiom itself and also on global factors such as the discourse history and potentially other triggers (for example, see the discussion relating to the configuration hypothesis in the introduction of Chapter 4). Given this perspective on idioms, NLP models that use relatively simple representations tend to struggle. However, despite this, the state-of-the-art in many idioms processing tasks restrict themselves to simple representations, be they: raw fixedness measures; global topic models; or dictionaries.

In this thesis we investigate the usage of more complex and flexible representations (in comparison to current approaches) to unpack the entangled set of idiom properties and reduce the amount of surrounding text (*i.e.*, discourse history) required to process such expressions. We argue that NLP systems should utilise more flexible and complex representations and better "history models" of discourse history to process idioms. We analyse these complex representation in a variety of idiom related tasks, ranging from problems where the discourse history can be discarded (*idiom type* identification) to tasks where the representation of discourse history is important and is either known ahead of time (*idiom token* identification), must be reconstructed after each input word (language modeling) or must be reconstructed after each output word (machine translation). We show that complex representations are necessary to capture the subtleties of idioms and reduce the amount of surrounding text required. Furthermore, the more complex representations improve the scalability of the models in terms of either reducing the size of the models (in terms of the number of parameters) or enabling the usage of a single classifier in *idiom token* identification (instead of a specific classifier for each expression).

1.1 Contributions

This thesis makes the following contributions over different tasks regarding idioms:

1. We present a quantification of the negative impact idioms cause in Statistical Machine Translation (SMT) systems and demonstrate that baseline systems are not able to easily translate idioms.

2. We demonstrate that simple statistical representations can be used to represent local context about idioms but improved results on *idiom type* identification are obtained through more complex representations where the model can utilize interactional terms between the features in the representation.

3. We show that by using complex distributed representations of a sentence containing a candidate idiom, the amount of surrounding text that an *idiom token* classifier needs to maintain its level of performance can be reduced from several paragraphs surrounding the sentence to the sentence itself.

4. We also show that by using these complex representations of a sentence containing a candidate idiom, a single classifier can be designed for *idiom token* classification in stark contrast with the current approach of training a different classifier for each expression.

5. In language modelling we use distributed representations and also

improve the systems' ability to access historic/global context by using an attention module. In such settings where the context is reconstructed after each input, such as in Language Models based on Recurrent Neural Network (RNN-LMs), the performance of such models in terms of perplexity is greatly improved in both idioms processing and regular language. We also demonstrate that the inclusion of the attention module helps the RNN-LM model to bridge long distant dependencies occurring in language.

6. We show that, in settings where the input context is known ahead of time but the output context is reconstructed after each prediction, that complex representations that merge both current and past context by means of an attention module help in bridging long distant dependencies in the output and improve idioms translation in some settings.

We believe these contributions will help to expand the frontiers of research involving idioms in NLP by pointing towards different directions for work beyond traditional statistical methods. In addition, we believe our contributions will impact on a range of other research fields, including (but not limited to) language modeling, neural networks and machine translation.

1.2 Summary and Structure

The main body of this thesis is structured as follows. Chapter 2 reviews SMT systems and outlines a set of experiments that we developed to measure the impact of idioms in SMT. Chapter 3 then moves on to present an alternative set of lexical fixedness metrics to overcome limitations found in current *idiom type* identification models and also demonstrates that higher dimensional representations yield improvements over simpler representations.

Chapter 4 details our investigation of distributed semantic representations to build a "general" classifier for *idiom token* identification. Building a general idiom token identification model (*i.e.*, a model that can do idiom token identification for many different idiomatic expressions) is a significant step beyond the current "per-expression" approach where a separate idiom token model is required to be trained for each idiom.

In Chapter 5 we explore the use of attention techniques to bring forward information about the past discourse history when reconstructing the distributed representations after each input in language modeling. Chapter 6 details our investigation into adapting the attention techniques proposed for language modeling to neural machine translation, a task where the input context is known ahead of time but where the output must be reconstructed after each output. Finally, Chapter 7 summarizes the key contributions of this work and highlights opportunities for

additional research.

1.3 Publications Arising from this Thesis

The publications that form the basis of this thesis are listed bellow:

- **Chapter 2.** Salton, G. D., Ross, R. J., and Kelleher, J. D. (2014a). An Empirical Study of the Impact of Idioms on Phrase Based Statistical Machine Translation of English to Brazilian-Portuguese. In *Third Workshop on Hybrid Approaches to Translation (HyTra) at 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 36–41

- **Chapter 3.** Salton, G. D., Ross, R. J., and Kelleher, J. D. (2017b). Idiom type identification with smoothed lexical features and a maximum margin classifier. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP'2017)*, pages 642–651

- **Chapter 4.** Salton, G. D., Ross, R. J., and Kelleher, J. D. (2016). Idiom token classification using sentential distributed semantics. In *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics*, pages 194–204

- **Chapter 5.** Salton, G. D., Ross, R. J., and Kelleher, J. D. (2017a). Attentive language models. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP*

2017)

Other relevant publications:

- In Salton et al. (2014b) we suggested a pre- and post-processing pipeline to process idioms in Statistical Machine Translation while keeping the system unchanged.

Chapter 2

Idioms and Statistical Machine Translation

It is well known in the Natural Language Processing (NLP) field that idioms pose problems for most NLP systems (Salton et al., 2014a). In this chapter we analyse the performance of Statistical Machine Translation (SMT), one of the most important Natural Language Processing (NLP) systems, when idioms are present in the input to these systems. We limit our investigation to the following question regarding idioms and SMT:

> *What is the impact of idioms on the performance of an SMT system?*

It is commonly accepted in the Machine Translation field that the performance of SMT systems degrades when the input sentence contains an idiom as they often try to translate the idiom as "literal text" (Vilar et al.,

2006). However, it is difficult to find in the literature a formal measure of that impact. In this chapter we address this gap in the literature.

We proceed with a review of SMT methods in Section 2.1; and then go on to present previous, relevant work regarding idioms and other multiword expressions (MWEs) in SMT in Section 2.2. Following this we then present the investigation of the question outlined above in Section 2.3; and, finally, we draw our conclusions in Section 2.4.

2.1 Statistical Machine Translation

As proposed by Brown et al. (1990), a SMT system is an application of the Bayes' Theorem:

$$arg\ max_T\ P(T|S) = arg\ max_T\ P(S|T) \times P(T) \tag{2.1}$$

where S is the input written in the "source language"; T is the translation of S written in the "target language"; $P(S|T)$ is the *Translation Model* (TM) which captures the probability of producing S given the translation T; and $P(T)$ is the *Language Model* (LM) which captures the probability of producing T in the target language. Note that, in the TM component, the position of S and T are inverted so, in effect, SMT is an application of Bayes' *Noisy-Channel* model. In other words, the

TM is derived in the opposite direction, from source to target.

Framed this way, a SMT system employs two distinct steps: training and decoding (Hearne and Way, 2011). In the *training* step, the TM component is estimated on a bilingual aligned corpora, *i.e.*, the text in the portion corresponding to the source language is aligned at sentence level to their translations in the portion corresponding to the target language. Also in the training step, the LM is estimated on monolingual data that is generally much larger than the bilingual data (Brown et al., 1993). Given a source sentence S to be translated, the *decoding* step involves a search for the best possible translation T allowed by the TM and the best possible reordering of the target text T given the LM.

Although the work of Brown et al. (1990) defines the fundamentals of the model, the current state-of-the-art in SMT is Phrase-based SMT (PBSMT) (Koehn et al., 2003). PBSMT extends the standard SMT word-by-word (*uni*-gram) approach by partitioning the source sentence into segments or "phrases" (*i.e.*, higher order *n*-grams). These segments are then translated into the target language using the TM (which in PBSMT is a Phrase-Table (PT)) and reordered using the LM, if needed. Note that, in the case of PBSMT systems, "phrases" are generally an arbitrary subsequence of *n*-grams without any linguistic motivation. In addition, PBSMT systems often employ a log-linear model (Och and Ney, 2002) that enables the use of an unlimited number of features in aggregation to the original noisy-channel model. The log-linear model

is defined as:

$$arg\ max_T\ P(T|S) = arg\ max_T \sum_{m=1}^{M} \lambda_m \cdot h_m(S,T) \qquad (2.2)$$

where $h_m(T,S)$ is the log-probability score assigned to a source-target pair by the m-*th* feature; and λ_m is the weight representing the relative importance of the m-*th* feature. The log-linear model with the features of the original word-by-word model of Brown et al. (1990) is represented as:

$$arg\ max_T\ P(T|S) = arg\ max_T\ \lambda_1 \cdot \log P(S|T) + \lambda_2 \cdot logP(T) \quad (2.3)$$

where $P(S|T)$ is the TM; $P(T)$ is the LM; and λ_1 and λ_2 are the weights for the TM and LM respectively. By applying the log-linear model, it is possible to "tune" the weights of each feature in the model for different translation tasks. Moreover, it is easy to add more features into this setting by simply including new features into the feature set (Hearne and Way, 2011) and assigning a weight (which can be tuned later) to that feature.

Although in their baseline form PBSMT systems limit themselves

to a direct translation of n-grams without any syntactic or semantic context, these systems are still the state-of-the-art in SMT. One consequence of these limitations however is that these systems do not model idioms explicitly (Bouamor et al., 2011).

2.2 Idioms in SMT

Constructions involving non-compositional MWEs, such as idioms, are problematic for SMT as these systems often make the assumption that text simply consists of word sequences with no semantics (or pragmatics) involved. Therefore, SMT systems do not make any distinction between literal and figurative meanings of MWEs. According to Cap et al. (2015), an SMT system with sufficient coverage of a particular idiom would be more likely to produce correct translations for that expression when its constituent words are adjacent but, otherwise, it is difficult for the system to produce the correct translation.

A problem related to the coverage of idioms is that, in human languages, idioms are very frequent as a class, given that native speakers rarely realize how often they employ such expression (Sag et al., 2002) but, however, they are not frequent in NLP resources such as in lexicons and grammars (Rondon et al., 2015).

Most work on translating non-compositional content is related to general MWEs, *e.g.*, collocations. Despite the fact collocations also display

semantic idiomacity, these are high-fixed MWEs that are characterized by their statistical idiomacity. Approaches designed for translation of collocations are more appropriate for high-fixed idioms as they often can be treated as *word-with-spaces* (Sag et al., 2002). However, most idioms are low-fixed and allow different types of variations, including words in between their original constituents and, consequently, methods designed for collocations will not work for many idioms.

Following this line of research on collocations, (Zhang et al., 2008) propose a method to avoid overfitting caused by the Expectation Maximization (EM) algorithm's bias towards longer sentences. That problem arises from the fact that the EM algorithm tries to maximize the likelihood of its training data and, therefore, prefers to "explain" a sentence pair with a longer and single phrase pair. The authors assign more probability mass to vectors of parameters with few non-zero values using Bayesian methods and, therefore, favour more frequent and shorter phrases. Although the authors show that their method improves the translation of collocations in the form of frequent *bi*-grams, the method is difficult to apply for the case of given how infrequent they are in corpora and the fact that idioms typically are composed of more than two words (*e.g.* vern-noun idiomatic combinations or VNICs, which are idioms composed by a direct verb and a noun in the direct object position). (Ren et al., 2009) propose a method to extract bilingual MWEs from a trained PBSMT system and add them to the corpora to retrain

the system and to a bilingual dictionary that is used as a feature to guide the search for the best translation. Nevertheless, as is often the case, the phrase aligner extracts the most common phrase alignments from the corpora to the PT and, therefore, enabling the system to capture collocations from corpora.

Along the same line of extracting bilingual collocations, Bai et al. (2009) develop a method to extract the MWEs and their translations from bilingual corpora based on their frequency and feed the extracted MWEs directly to the decoder once the expression is recognized. In order for this method to work, the MWE in the input must match directly one of the MWEs in the bilingual dictionary and, therefore, is more suited to the translation of collocations as they do not have changes in word order.

On a different line of research but also focusing on collocations, Xiong et al. (2010) propose a method to predict the boundaries of a MWE in which two different classifiers are trained: one classifier to predict where the MWE begins; and the other to predict where the MWE ends. Therefore, these boundary predictions require the MWE, after the boundaries are predicted, to be translated as one single unit following a *word-with-spaces* approach.

Focusing on monolingual collocations, Bouamor et al. (2012) developed a method to extract monolingual MWEs from the source side of a bilingual corpus and then apply vector space models to extract the

candidate translations from the target side of the same bilingual corpus. The authors conduct different experiments using these bilingual dictionaries: they retrained their PBSMT systems including these MWEs in the bilingual corpus; used it as a feature to indicate whether a phrase was a MWE or not; and tried to insert this dictionary directly into the system's phrase-table. Using either of the three methods, the authors report improvements on the translation of collocations.

In Bungum et al. (2013) the authors present an approach to find MWEs in bilingual corpora based on word sense disambiguation techniques. Their assumption is that a word, when in a semantically non-compositional MWE, has a different meaning than its original one and this feature can be used to identify the enclosing MWE. The authors then extract bilingual MWEs to a dictionary and use this dictionary identify MWEs in the source language. However, for their method to work, a MWE in the source language must match *word-by-word* a dictionary entry during the translation process.

Investigating a different set of MWEs other than collocations, Kordoni and Simova (2014) present an evaluation regarding the impact of the verb constituents of MWEs in the form of phrasal verbs when obtaining word alignments for SMT systems and the impact on the output. Although the authors demonstrated that extracting phrasal verbs by using a separate classifier and including these phrasal verbs into the phrase-table improve the translation of such MWEs, the authors limit the study

to English MWEs that can be translated to a single word in Bulgarian.

Despite these previous works on translating more general MWEs in PBSMT, to our knowledge there is little work specific to idioms. Okita (2011) discusses possible techniques that can be used to extract linguistic domain knowledge in order to improve word alignments when idioms are present. Although the authors addresses idioms these expressions are not the main focus of the work. In Ho et al. (2014) the authors describe the process they use to build a bilingual idiom dictionary and integrate it into a Chinese to English PBSMT pipeline as part of the PT component. However, as demonstrated by other authors (*e.g.*, Okuma et al. (2008)), introducing a dictionary into a PT is problematic for the cases when the MWE is not high-fixed and, in general, introduce errors in the translation of literal language.

More recently, Cap et al. (2015) proposed a method where they tag the verbs that belong to an idiomatic construction using a "special tag" while they tag the same verbs occurring in literal constructions with a different tag. They then use this tagged corpora to train an SMT system following the intuition that the system would be able to distinguish the correct translations for the expressions based on the tagged verbs since the system would learn when to choose each version of the tagged verbs. One limitation of this approach, however, is that the PBSMT system must be trained over the tagged corpora meaning that it must be pre-processed prior to training (*i.e.*, the idioms must be identified be-

forehand) and, for each new idiom that was not in the original set, the system must be re-trained. In addition, another limitation of this approach is that after the system has been deployed all the sentences that are input into the system for translation must also be pre-processed and tagged in a similar manner.

Finally, in the context of this thesis there are a number of other shortcomings of the above work on idioms in machine translation. First, none of these works explicitly evaluate the impact of idioms on machine translation, for example by comparing the BLEU score of sentences containing and not containing idioms. To the best of our knowledge, there is no work focusing only on the translation of figurative content in the Neural Machine Translation field.

2.3 Measuring the Impact of Idioms on SMT

In this section we gauge the impact of idioms on a PBSMT system. In order to measure this impact we ran an experiment that compared the BLEU scores (Papineni et al., 2002) of a PBSMT system when tested on three distinct bilingual corpora. Two of these test corpora consisted of sentences containing figurative usages of idioms that can also be used literally and the third corpus consisted of sentences with only literal language[1]. By comparing the BLEU score of the PBSMT system on each of

[1]In this corpus we ensured that it did not contained any literal usage of idioms or other forms of figurative language

these corpora we hope to gauge the size of the research problem idioms pose to those systems.

The description of the experiment is organized as follows: Section 2.3.1 describes the design and creation of the corpora used in the experiments; and Section 2.3.2 both presents the experiment and reports the results. Finally, we discuss the results in Section 2.3.3.

2.3.1 Datasets

The experiment we describe in this section had two direct targets: (a) we wish to quantify the effect of idioms on the performance of a PBSMT system; (b) we hope to better understand the differences (if any) between high-fixed and low-fixed idioms with respect to their impact on PBSMT systems.

PBSMT training requires a paired language corpus for the targeted language pair - generally the larger the corpus the better. Unfortunately, for most language pairs a large bilingual corpus does not exist and, if the corpus exists, it often requires a paid license to use (Tang, 2012). Consequently, for this experiment we created our own corpora.

We created a training corpus by concatenating the English/Brazilian--Portuguese part of the Fapesp-v2 corpus (Aziz and Specia, 2011) with a series of other corpora available made available by the OPUS Project (Tiedemann, 2012) for that language pair. The resources used in this

Table 2.1
Statistics of corpora used in the idioms' impact experiments. EN stands for English and PT-BR for Brazilian-Portuguese, following the Internet Engineering Task Force (IETF) language codes (Phillips and Davis, 2009).

Corpus	Sentences	EN Tokens	PT-BR Tokens
Fapesp-v2	0.15M	4.3M	4.0M
OpenSubtitles2013	37.1M	298.7M	255.8M
KDE4	0.2M	2.4M	2.6M
PHP	40.4K	0.5M	0.2M

step were the OpenSubtitles2013 corpus[2], the PHP Manual Corpus[3] and the KDE4 localization files (v.2)[4]. No special tool was used to clean these corpora prior to concatenation and the files were compiled as is. Table 2.1 presents statistics for these corpora.

The Fapesp-v2 corpus is based on scientific texts and, thus, is unlikely to have a high frequency of idioms. Thus, the main reason for the concatenation is the fact that, from the selected corpora, only the Fapesp-v2 is professionally translated and, thus, the PBSMT can extract good quality "phrases" from that corpus.

To account for idiomatic content, we also consider a different corpus type. As pointed out by Hardmeier and Volk (2009), a good subtitle must help the viewer to quickly understand the dialogue in the screen without distracting them from the video. Consequently, subtitle text is often "condensed" and likely to contain MWEs. Therefore, in order to enable the system to account for more figurative language we also included the subtitles corpus. However, using a subtitles corpus has its

[2]http://opus.lingfil.uu.se/OpenSubtitles2013.php
[3]http://opus.lingfil.uu.se/PHP.php
[4]http://opus.lingfil.uu.se/KDE4.php

own challenges. The first issue to consider is that subtitles are a specific register of language which make them different from either written texts or spoken dialogue (Biber et al., 1998). The second issue is the inherent noise in these corpora resulting from the subtitles being "condensed" and, thus, presenting a different word alignment between the original text and the translated text (due to screen restrictions) (Hardmeier and Volk, 2009). In addition, most freely available subtitles are translated by volunteers and may include linguistic mistakes of various levels (Petukhova et al., 2012). Thus, subtitle corpora are often likely to be imperfect translations.

The focus of this experiment was to estimate the impact of the presence of idioms on PBSMT systems. There are, however, many factors that can affect the performance of a PBSMT system on a given corpus and these could act as confounding factors in our analysis unless they are controlled for. One such factor is the variation in sentence length across the corpus. In order to control for this factor we filtered the training corpus so that it only contained sentences within the range of [15, 20] words. The motivation for using this range was that it contained the most common lengths in the corpus. After removing the sentences outside the specified range, the corpus used for training consisted of 17,288,109 pairs of sentences (approximately 50% of the initial collected corpus), with another 34,576 pairs of sentences (also within the range $[15, 20]$) sampled for the "tuning" process.

As idioms are a heterogeneous class, we decided to focus on expressions formed from the combination of a verb and a noun as its direct object (*e.g.*, *hit+road* and *lose+head*). These expressions are called verb+noun combinations or VNICs and they are the most frequent class of idioms in English (Villavicencio and Copestake, 2004). VNICs are also notable for their cross-lingual occurrence and high variability, both lexical and semantic (Baldwin and Kim, 2010). In addition, it is possible for a particular verb+noun combination to have both idiomatic and literal usages and these usages must be distinguished if an NLP system is to process a sentence appropriately. In addition, they present varying degrees of fixedness, *i.e.*, they allow different degrees of variation and can be classified into high-fixed and low-fixed idioms.

Given the fact that there is no readily available bilingual datasets of idiomatic expressions to test English/Brazilian-Portuguese, we had to manually build our test sets of high-fixed and low-fixed idioms (called "*High-idiomatic* Corpus" and "*Low-idiomatic* Corpus" respectively). In order to do that, we took the list of 17 high-fixed English verb+noun idioms presented by Fazly et al. (2009), and used it to build the "*High-idiomatic* Corpus". This corpus consisted of 170 sentences (within the range of $[15, 20]$ words) containing idiomatic usages of these idioms, 10 sentences per idiom in the list. These English sentences were collected from websites of idiom dictionaries[5] (which contain sample sentences

[5]As we wanted to collect 10 sentences for each VNIC and due to the limitations of sentence length (15 to 20 words) we were not able to collect the "*High-idiomatic* Corpus" and the "*Low-idiomatic*

containing usages of such idioms) and manually translated into Brazilian-Portuguese. After that step, the translations were then checked (also manually) by a second human translator.

To build the "*Low-idiomatic* Corpus", we used the list of 11 low-fixed English verb+noun idioms, *e.g. get+wind*, also presented in Fazly et al. (2009). This corpus consisted of 110 sentences (within the range of $[15, 20]$ words) containing idiomatic usages of these idioms, 10 sentences per idiom in the list. These English sentences were also collected from websites of idiom dictionaries and manually translated into Brazilian-Portuguese. Once again, after the translation, the corpus was also manually checked by a second human translator. Table 2.2 presents the English high-fixed idioms combinations used in this experiment and their Brazilian-Portuguese translations. Table 2.3 presents the English low-fixed idioms combinations used in this experiment and their Brazilian-Portuguese translations.

As we are analysing the impact of idioms in a PBSMT system we must also measure the performance of the system on literal language, without figurative content similar to idioms. With that in mind, the "Clean Corpus" was built. It consisted of 850 sentences with their translations and was created by sampling sentences in the same length range ($[15, 20]$ words) as those in the training corpus. These sentences were then removed from the training corpus. Given that the initial corpus was

Corpus" from the training corpus. Thus, the samples were collected from the Internet.

Table 2.2
The English high-fixed idioms used in the experiment and their Brazilian-Portuguese Translations. The idioms marked with an * have direct translations of their constituents resulting in a MWE with the same idiomatic meaning in Brazilian-Portuguese (these idioms are presented between double quotes and italics). Also, note that not all translations results in a verb+noun idiom in the target language.

English	Brazilian-Portuguese
blow+top	perder+paciência
blow+trumpet	"*gabar-se*"
cut+figure	causar+impressão
find+foot	"*adaptar-se*"
get+nod	"*obter permissão*"
give+sack	"*ser demitido*", "*demitir*"
have+word	ter+conversa
hit+road	"*cair na estrada*"
hit+roof	"*ficar zangado*"
kick+heel	"*deixar esperando*"
lose+thread	"*perder o fio da meada*"
make+face*	fazer+careta
make+mark	deixar+marca
pull+plug	"*cancelar algo*"
pull+punch	"*esconder algo*"
pull+weight	"*fazer sua parte*"
take+heart	"*ficar confiante*"

created from the union of corpora from different domains, the "Clean Corpus" was randomly split into 5 datasets containing 170 sentences each in an attempt to reduce the specific influence of any of those domains on the BLEU score. We called these "Clean1" to "Clean5". Special care was taken to not have any idioms or any other form of figurative language in any of these five clean test corpora.

To run the experiment and perform the measurements, we trained a PBSMT system for the English/Brazilian-Portuguese language pair using Moses toolkit (Koehn et al., 2007) following its "baseline" settings (Koehn et al., 2008).

Table 2.3
The English low-fixed idioms used in the experiment and their Brazilian-Portuguese Translations. The idioms marked with an * have direct translations of its constituents resulting in a MWE with the same idiomatic meaning in Brazilian-Portuguese (these idioms are presented between double quotes and italics). Also, note that not all translations results in a verb+noun idiom in the target language.

English	Brazilian-Portuguese
blow+whistle	"*botar a boca no trombone*"
get+wind	ouvir+murmúrios
hit+wall	"*dar de cara num muro*"
hold+fire	"*conter-se*"
lose+head*	perder+cabeça
make+hay	dar+graças
make+hit	fazer+sucesso
make+pile	fazer+grana
make+scene*	fazer+cena
pull+leg	pegar+pé
see+star*	ver+estrela

2.3.2 Experiments and Results

Table 2.4 lists the BLEU scores for each of the test corpora containing idioms ("*High-idiomatic* Corpus" and "*Low-idiomatic* Corpus") and the average BLEU scores over the clean corpora.

The differences among the BLEU scores for the clean corpus and both idiomatic corpora indicate that English VNICs pose a significant challenge to baseline PBSMT systems. On the corpora containing idioms the PBSMT system achieved only half of the average score ob-

Table 2.4
BLEU scores calculated for the "*High-idiomatic* Corpus", "*Low-idiomatic*" and the average for the "Clean Corpus".

Corpus	BLEU
High-idiomatic	23.12
Low-idiomatic	24.55
Clean (average)	46.28

Table 2.5
BLEU scores for individual expressions in the "*High-idiomatic* Corpus".

Expression	BLEU Score
blow+top	22.08
blow+trumpet	19.38
cut+figure	20.15
find+foot	24.36
get+nod	22.06
give+sack	23.03
have+word	20.91
hit+road	24.53
hit+roof	21.34
kick+heel	18.85
lose+thread	21.81
make+face*	28.62
make+mark	29.46
pull+plug	19.71
pull+punch	28.34
pull+weight	19.94
take+heart	23.41

tained for literal language ("Clean Corpus").

The second question that we examined in the experiment was whether there was a difference in performance between the high-fixed and low-fixed idioms. Table 2.5 lists the BLEU scores for each of the high-fixed verb+noun combinations used in the "*High-idiomatic* Corpus" and Table 2.6 lists the BLEU scores for each of the low-fixed verb+noun combinations from the "*Low-idiomatic* Corpus". Figure 2.1 display a box plot of the BLEU scores obtained for individual expressions on both high-fixed and low-fixed corpora.

From Tables 2.5 and 2.6 and from Figure 2.1 we note that there is a difference in the BLEU score obtained by the SMT for both idiomatic

Table 2.6
BLEU scores for individual expressions in the "*Low-idiomatic* Corpus".

Expression	BLEU Score
blow+whistle	17.75
get+wind	19.06
hit+wall	16.52
hold+fire	23.26
lose+head*	37.40
make+hay	15.87
make+hit	25.48
make+pile	25.31
make+scene*	36.93
pull+leg	15.90
see+star*	37.86

corpora. Although we found a statistical difference[6] between the results of the Clean and the *High-idiomatic* corpora and between the results of the Clean and *Low-idiomatic* corpora ($p \ll 0.05$), we found that there is almost no statistical difference ($p = 0.85$) between the results of *High-idiomatic* and *Low-idiomatic* corpora.

2.3.3 Discussion

One possible confounding factor in our results is the fact that both idiomatic corpora were sampled from a different distribution than the training corpora. Although we acknowledge this limitation, we also expect that the trained PBSMT system was able to extract some MWEs into its PT component given that the subtitles corpora used to train the system is likely to contain such expressions. Therefore, even though the training and the idiomatic test corpora do not come from the same distribution,

[6]We calculated the statistical significance using the method of Koehn (2004).

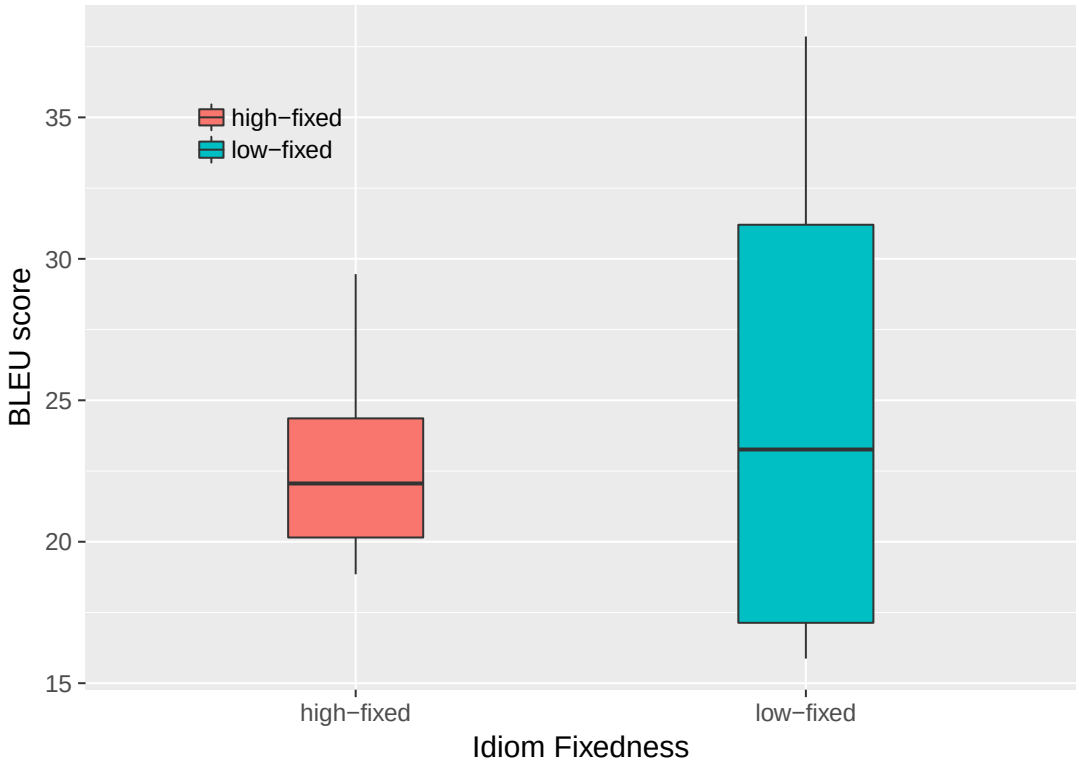


Figure 2.1
Box plot of BLEU scores obtained by the baseline SMT system on individual expressions in both high-fixed and low-fixed corpora.

the PBSMT system should be able to handle MWEs.

Interestingly, some idioms have a direct translation (*i.e.* a word-to-word translation) from English to Brazilian-Portuguese that the baseline PBSMT system could handle correctly. However, the BLEU scores were still lower than those of the literal language (see the expressions marked with a * in Table 2.2 and Table 2.3).

On the differences between high-fixed and low-fixed idioms, despite the fact that the Low-fixed corpus had more variation across the results for individual expressions (given that they allow more lexical and syntactic variations), we believe the absence or statistical difference indicates that both types of VNICs pose the same problems to PBSMT.

2.4 Conclusions

In some respects the results of this experiment are not surprising: it is generally accepted that the presence of idioms can negatively affect the performance of SMT systems. However, empirically measuring the extent of the impact of idioms on SMT system performance is novel and, furthermore, the magnitude of this estimated impact (nearly halving the systems' BLEU score) is somewhat surprising. It is evident that the problem in translating idioms has not been solved using a baseline PBSMT system. Given the magnitude of the problem posed by idioms to PBSMT systems, it is likely that there is no simple solution to this challenge.

Although BLEU scores are generally dependent on the training and test corpora, it is worthwhile having a quantification of the potential issues that idioms pose for PBSMT. In addition, these results are a starting point to develop a methodology to process idioms for MT. It is thus a central argument in this thesis that idioms must necessarily be explicitly modeled within NLP systems in order for NLP systems to adequately handle the nuances of idiom processing. In order to do this we need to better capture the nature of idioms as phenomena in language. In the next chapter we turn to that task.

Chapter 3

Idiom type Identification

The correct handling of idioms in NLP systems often requires idiom dictionaries. An example can be seen in the field of SMT, where most current systems rely on large repositories of idioms to process these expressions (Peng et al., 2014). However, generating and maintaining these dictionaries is challenging as new idioms are created on a daily basis (Fazly et al., 2009) and, therefore, compiling a dictionary by hand is expensive and time consuming. NLP systems need reliable ways of automatically identifying idioms to keep their idiom resources up-to-date (Bannard, 2007). This is particularly important for under-resourced languages where no, or very limited, idiom dictionaries may yet exist.

A less expensive method of building idiom dictionaries is to apply *idiom type* identification to a selected corpora, in the desired language, to extract idioms. In our research we focus on the task of identifying *idiom types* composed of a transitive verb and a noun occurring in its direct object position that have an idiomatic meaning associated (Nun-

berg et al., 1994). These expressions are referred to as VNICs - short for *verb+noun idiomatic combination*. VNICs are the most frequent class of idioms (Villavicencio et al., 2004) and occur across almost all languages (Baldwin and Kim, 2010). This class of idioms has been extensively studied and is known to exhibit varying degrees of syntactic and lexical fixedness and semantic non-compositionality. We call the task of *idiom type* identification focused on VNICs as *VNIC type* identification.

Within the domain of *VNIC type* identification, for many years the state-of-the-art method was the work of Fazly et al. (2009), who devised a set of fixedness metrics based on the observation that idioms are generally more lexically and syntactically fixed in comparison to non-idiomatic verb+noun combinations. Nevertheless, we believe Fazly's et al. approach is prone to limitations and has limited results when applied to small corpora, as the case for under-resourced languages.

In this chapter we present an approach to *idiom type* identification based on blending a rich set of fixedness metrics with a non-linear classifier model to produce a new state-of-the-art for *VNIC type* identification. We begin this process by reviewing previous work on *idiom type* identification in Section 3.1, including the current state-of-the art in *VNIC type* identification and discussing the limitations imposed by particular choices of that model. In Section 3.2 we propose modifications to overcome these limitations and evaluate the metrics to extract VNICs from corpora in Section 3.3.

We also assess the goodness of the metrics for *VNIC type* classification when classifying verb+noun pairs into VNICs and non-VNICs in Section 3.4. Following that, we show that using the fixedness metrics as input features to a Support Vector Machine (SVM) classifier results in an improved method for *VNIC type* identification in comparison to Fazly's et al. model in Section 3.5. Finally, we discuss the results in Section 3.6 and draw our conclusions in Section 3.7.

3.1 Previous Work on *Idiom type* Identification

Idiom type identification refers to the task of identifying expressions that can potentially be used as idioms. A successful *idiom type* classifier would identify expressions such as *shoot the breeze* ("have a casual conversation") and *make the cut* ("come up to a required standard") as potentially having idiomatic interpretations associated with some instances of their use, and other expressions (for example "make the coffee") as having a very low probability of being used in an idiomatic way.

Within the field of *idiom type* identification, researchers have proposed both supervised methods that rely on manually encoded knowledge and unsupervised methods that rely on knowledge automatically extracted from corpora.

On the supervised approaches, both Copestake et al. (2002) and Villavicencio et al. (2004) proposes methods based on formal grammars

to identify idiom types and other, more general, types of MWEs. It is worth mentioning the fact that, to date, the supervised methods strive to capture relevant information regarding the properties of specific idioms and thus, do not generalize beyond expressions known ahead of time. This is because they generally rely on hand-crafted rules which must be specifically designed for each particular expression (for example, by inspecting violation in selectional preferences).

Among the unsupervised approaches, Lin (1999) analyses the deviation of mutual information between two phrases to detect non-compositional constructions. Although designed to detect non-compositionality by means of measuring strength of lexical associations, this approach is designed to identify collocations and not idiom types. Bannard (2007) investigates the variability in the syntax of English idioms as a feature for identifying idiom types. Building upon the strength of lexical association and syntactic variability, Fazly et al. (2009) present a set of fixedness metrics (lexical and syntactic) designed to identify pairs consisting of a transitive verb and a noun in its direct object position (which we refer to as verb+noun pairs) that have an associated idiomatic expression (verb+noun idiomatic constructions or VNICs). Fazly's et al. approach to VNIC type identification is to assign to each verb+noun pair in the corpus an overall fixedness score which is a linear combination of a syntactic and a lexical fixedness metric. More recently, although not the state-of-the-art in VNIC type identification in English, Senaldi et al.

(2016) present a model using distributed semantics to identify idiom types in Italian. The authors analysed the differences between idiomatic and literal phrases in embedding spaces, in a similar fashion to lexical fixedness.

3.1.1 Fazly's Fixedness Model

Of all the previous work, probably the most complete and relevant to our needs here is the work of Fazly et al. (2009). The authors present a set of fixedness metrics designed to identify verb+noun pairs that have an associated idiomatic expression. The authors devise an overall fixedness score based on the evidence that idioms are more syntactically and lexically fixed than literal constructions

Of all the previous work, probably the most complete and relevant to our needs here is the work of Fazly et al., in the following subsections we present and analyze the details of this approach: syntactic fixedness (Section 3.1.1.1); lexical fixedness (Section 3.1.1.2); and the linear combination of these two fixedness metrics into the overall fixedness metric (Section 3.1.1.3). We finish with a discussion about these metrics (Section 3.1.1.4).

3.1.1.1 Baseline Syntactic Fixedness

Fazly et al. (2009) propose a metric to capture the syntactic fixedness of idioms based on the observation of Riehemann (2001) that idiomatic ex-

Table 3.1
Syntactic patterns: the verb *v* can be in active (v_{active}) or passive ($v_{passive}$) voice; the determiner (DET) can be NULL, indefinite (*a/an*), definite (*the*), demonstrative (DEM), or possessive (POSS); and the noun *n* can occur in singular ($n_{singular}$) or plural (n_{plural}).

No.	Verb	Determiner	Noun
pt_1	v_{active}	DET:*NULL*	$n_{singular}$
pt_2	v_{active}	DET:*a/an*	$n_{singular}$
pt_3	v_{active}	DET:*the*	$n_{singular}$
pt_4	v_{active}	DET:*DEM*	$n_{singular}$
pt_5	v_{active}	DET:*POSS*	$n_{singular}$
pt_6	v_{active}	DET:*NULL*	n_{plural}
pt_7	v_{active}	DET:*the*	n_{plural}
pt_8	v_{active}	DET:*DEM*	n_{plural}
pt_9	v_{active}	DET:*POSS*	n_{plural}
pt_{10}	v_{active}	DET:*other*	$n_{singular,plural}$
pt_{11}	$v_{passive}$	DET:*any*	$n_{singular,plural}$

pressions are expected to appear more frequently under their canonical syntactic form than literal combinations. The authors describes three types of syntactic variations that can be characteristic of idiomatic combinations: "Passivization" (due to the non-referential status of the noun in the idiomatic expression); "Determiner type" (which is affected by the semantic properties of the noun); and "Pluralization" (also due to the non-referential status of the noun in such constructions). Merging these three syntactic variations, a set $\mathcal{P}$ of eleven syntactic patterns, as presented in Table 3.1, can be obtained.

The goal of the baseline syntactic fixedness is to compare the behaviour of a target verb+noun pair to the behaviour of a "typical" verb+noun pair. The syntactic behaviour of a "typical" verb+noun pair is defined as a prior distribution over the set $\mathcal{P}$ and is calculated individu-

ally for each pattern $pt \in \mathcal{P}$ as follows:

$$\begin{aligned} P(pt) &= \frac{\sum_{v_i \in \mathcal{V}} \sum_{n_i \in \mathcal{N}} f(v_i, n_i, pt)}{\sum_{v_i \in \mathcal{V}} \sum_{n_i \in \mathcal{N}} \sum_{pt_k \in \mathcal{P}} f(v_i, n_i, pt_k)} \\ &= \frac{f(*, *, pt)}{f(*, *, *)} \end{aligned} \tag{3.1}$$

where pt is the pattern we are calculating the prior of; $f(*, *, pt)$ is the frequency of the pattern irrespective of any verb or noun; and $f(*, *, *)$ is the frequency of all $pt \in \mathcal{P}$, irrespective of any verb or noun.

The syntactic behaviour of the target verb+noun pair is defined as a posterior distribution over the set $\mathcal{P}$ given the pair's constituents. Thus, its syntactic behaviour is calculated for a pattern $pt \in \mathcal{P}$:

$$\begin{aligned} P(pt|v, n) &= \frac{f(v, n, pt)}{\sum_{pt_k \in \mathcal{P}} f(v, n, pt_k)} \\ &= \frac{f(v, n, pt)}{f(v, n, *)} \end{aligned} \tag{3.2}$$

where pt is the pattern we are calculating the posterior of; $f(v, n, pt)$ is the frequency of the target verb+noun pair $\langle v, n \rangle$ occurring under the syntactic pattern pt; and $f(v, n, *)$ is the frequency of the target verb+-noun pair occurring under any of the patterns $pt \in \mathcal{P}$.

The difference between the behaviour of the target verb+noun pair

and the "typical" verb+noun pair is calculated as the divergence between the posterior and the prior distributions over $\mathcal{P}$ as measured using the Kullback-Leibler divergence:

$$\begin{aligned}\mathcal{F}_{syn}(v,n) &= D(P(pt|v,n)||P(pt)) \\ &= \sum_{pt_k \in \mathcal{P}} P(pt_k|v,n) \log \frac{P(pt_k|v,n)}{P(pt_k)} \end{aligned} \tag{3.3}$$

where $P(pt|v,n)$ is the pattern posterior; and $P(pt)$ is the pattern prior.

Syntactic fixedness takes values in the range $[0, +\infty]$ where larger values denote higher degrees of syntactic fixedness, *i.e.*, where the verb+noun pair is more likely to be a VNIC.

3.1.1.2 Baseline Lexical Fixedness

Typically, idioms do not have lexical variants and, when they do, these variants are generally unpredictable (Fazly et al., 2009). Therefore, it is assumed that a verb+noun pair is lexically fixed (and likely to be a VNIC) to the extent that replacing one of its constituents by a semantically similar word does not generate another valid idiomatic combination. Based on this, lexical fixedness was proposed to capture the degree to which a given verb+noun pair is lexically fixed with respect to the set of its variants. These variants are generated by replacing either the verb or the noun by a word from a set of semantically similar words and is defined as:

$$S_{sim}(v,n) = \{\langle v_i, n\rangle | 1 \leq i \leq K_v\} \cup \{\langle v, n_j\rangle | 1 \leq j \leq K_n\} \quad (3.4)$$

where $\{\langle v_i, n\rangle | 1 \leq i \leq K_v\}$ is the set of similar combinations generated by replacing the verb by a word from the set of the K_v most similar words to that verb according to a thesaurus (while the noun is kept unchanged); and $\{\langle v, n_j\rangle | 1 \leq j \leq K_n\}$ is the set of similar combinations generated by replacing the noun by a word from the set of the K_n most similar words to that noun also according to a thesaurus also (while the verb is kept unchanged).

To measure the strength of the association between the target verb+-noun pair's constituents, Pointwise Mutual Information (PMI) (Church et al., 1991) is applied to the pair and to its set of similar combinations $S_{sim}(v,n)$.

$$\begin{aligned} PMI(v,n) &= \log \frac{P(v,n)}{P(v)P(n)} \\ &= \log \frac{N_{v+n} f(v,n)}{f(v,*)f(*,n)} \end{aligned} \quad (3.5)$$

where (v,n) is the verb+noun pair we are calculating the PMI of (be it the target pair or one of its similar combinations); N_{v+n} is the total

number of verb+noun pairs in the corpus; $f(v,n)$ is the frequency of the verb+noun pair occurring together; $f(v,*)$ is the frequency of the verb v occurring with any noun as its direct object in the corpus; and $f(*,n)$ is the frequency of the noun n occurring as a direct object of any transitive verb in the corpus.

The idea behind lexical fixedness is that the target verb+noun pair $\langle v,n\rangle$ is lexically fixed, and likely to be a VNIC, to the extent its PMI is larger then the mean PMI of the set $S_{sim}(v,n) \cup \langle v,n\rangle$. Following this assumption, lexical fixedness of a verb+noun pair is calculated as a standard z-score:

$$\mathcal{F}_{lex}(v,n) = \frac{PMI(v,n) - \overline{PMI}}{s} \tag{3.6}$$

where $PMI(v,n)$ is the PMI of the target verb+noun pair $\langle v,n\rangle$; $\overline{PMI}$ and s are the mean and standard deviation of PMI applied to the verb-+noun pairs listed in $S_{sim}(v,n) \cup \langle v,n\rangle$. Lexical fixedness falls into the range of $[-\infty,+\infty]$, where higher values mean higher degrees of lexical fixedness and that the verb+noun pair is likely to be a VNIC.

3.1.1.3 Baseline Overall Fixedness

As pointed out by a number of linguistic studies, a particular VNIC is expected to be both lexically and syntactically fixed to a greater extent then literal combinations (Fazly et al., 2009). To capture this overall fixedness of a verb+noun pair, both lexical and syntactic metrics are

merged in a weighted linear combination[1], as follows:

$$\mathcal{F}_{over}(v,n) = \omega\mathcal{F}_{syn}(v,n) + (1-\omega)\mathcal{F}_{lex}(v,n) \tag{3.7}$$

where the weight ω controls the relative contribution of each measure for predicting the VNIC's idiomacity. Overall fixedness takes its values in the range $[0,1]$, where values close to 1 mean higher degrees of overall fixedness. Therefore, if a verb+noun pair receive a score higher than a threshold it is classified as a VNIC and, in their work, Fazly et al. set the median score of the test set as the threshold to classify the verb+noun pairs into VNICs and non-VNICs.

3.1.1.4 Discussion

Fazly et al. have shown their fixedness model to be useful in VNIC type identification. However, their model does have limitations. The definition of lexical fixedness is based on PMI which is known to be biased towards infrequent events (Turney and Pantel, 2010). This property of PMI may lead to undesired results when computing lexical fixedness using counts obtained from a corpus. Furthermore, it is difficult to interpret PMI for those pairs listed in $S_{sim}(v,n)$ that are not observed in the corpus. All pairs are generated by using synonyms (or at least similar related words) and should be acceptable combinations in language.

[1]Note, lexical and syntactic fixedness fall into a different range of values so we first rescale them to the range $[0,1]$ before combining them.

Therefore, the pairs's components should carry information about each other, even if it is small. Thus, we see that just discarding the pairs[2] would affect the result and reduce the power of the model. Therefore, especially for under-resourced languages with small corpora (where the chance of many pairs in $S_{sim}(v, n)$ not being observed is high), a more efficient way to measure the pair's association strength is needed.

3.2 Alternative Lexical Fixedness

Given the difficulties identified at the end of the last section, in the following we propose a number of new models which provide a more solid basis for measuring lexical fixedness. We proceed by describing and motivating these metrics in Sections 3.2.1 to 3.2.4. Then in Section 3.2.5 we show the combination of baseline syntactic fixedness with each lexical fixedness metric (baseline lexical and one of our proposed metrics) to form the overall fixedness metrics in Section 3.2.5.

3.2.1 Lexical Probabilities

As previously highlighted, a VNIC is expected to occur more frequently in language than its semantically similar variants. From a probabilistic perspective, we can assume that a VNIC has a higher probability of occurring in language than its semantically similar variants (*e.g.*, literal

[2]In other words, discarding the pairs is the same as setting PMI = 0, following *Information Theory* conventions.

variants). Following this intuition, we first propose to replace the PMI of a target verb+noun pair by the pair's probability estimated from the corpus as a base for a lexical fixedness metric. The probability for a verb+noun pair is estimated as:

$$P(v,n) = \frac{f(v,n)}{f(*,*)} \tag{3.8}$$

where $f(v,n)$ is the frequency of the verb+noun pair in a direct object relation (be it the target verb+noun pair or one of its similar variants), occurring in the corpus; and $f(*,*)$ is the frequency of all verb+noun pairs that occur in a direct object relationship in the corpus.

Similar to the baseline, we also assume that a target verb+noun pair $\langle v,n \rangle$ is lexically fixed, and likely to be a VNIC, to the extent its probability of occurring in language deviates positively from the mean probability of the set $S_{sim}(v,n) \cup \langle v,n \rangle$. Therefore, we can also calculate lexical fixedness based on a z-score:

$$\mathcal{F}_{lex} = \frac{P(v,n) - \overline{P}}{s} \tag{3.9}$$

where $P(v,n)$ is the probability of the target verb+noun pair; and $\overline{P}$ and s are the mean and standard deviation of P applied to $S_{sim}(v,n) \cup \langle v,n \rangle$.

3.2.2 Smoothed Lexical Probabilities

While the metric above is likely to be an improvement on PMI-based measurements, a lexical fixedness metric using raw probabilities may have some of the same disadvantages as a PMI-based metric when estimated with counts from a corpus. For example, when a verb+noun pair listed in $S_{sim}(v, n)$ does not occur in the corpus we end up with a probability of zero. Of course, just because a particular verb+noun pair does not appear in a corpus, does not mean that this combination cannot occur in language at all!

Inspired by the use of smoothing techniques in language modelling research to overcome the problem of unseen *n*-grams with zero probabilities, for our second metric we cast a VNIC as a *bi*-gram composed of a verb+noun pair ignoring the words in between the verb and the noun. This framing allows us to apply *Modified Kneser-Ney smoothing* to the probabilities of our verb+noun pairs thereby ensuring that all verb+noun pairs in our experiments (including unseen variants listed in $S_{sim}(v, n)$) have non-zero probabilities[3]:

$$P_{smoothed}(v, n) = Modified\ Kneser - Ney[P(v, n)] \tag{3.10}$$

[3]There are a number of (simpler) smoothing techniques that we could have used, such as add-1 or Laplace smoothing. However, we chose *Modified Kneser-Ney smoothing* as it is considered the state-of-the-art smoothing technique for *n*-grams (Brychcín and Konopík, 2014).

As with the lexical fixedness metric based on probabilities, we stick with the same assumption regarding how likely a verb+noun pair is to be a VNIC. Thus, the lexical fixedness based on smoothed probabilities is calculated using a z-score as in Equation (3.9).

3.2.3 Interpolated Lexical Probabilities

When estimating a language model from small corpora, we may suffer with occurrences of outliers or under-representative samples of *n*-grams (Koehn, 2010). This problem happens if the higher-order *n*-grams are too sparse and, thus, may be unreliable. The problem is more common when small corpora are used to estimate the model. As we are now considering the verb+noun pair as a *bi*-gram we may also have to face this problem.

To overcome these difficulties, it is a common practice to rely on lower-order *n*-grams, which are considered more robust, even if the higher-order *n*-gram have been observed. To do that, one can simply interpolate the high- and low-order *n*-grams into a single probability and, thus, bring together the benefits of longer contexts in higher-order *n*-grams and the robustness of low-order *n*-grams.

A simple but efficient way to interpolate higher- and lower-order *n*-grams is to first apply *Modified Kneser-Ney smoothing* and then sum the smoothed probabilities. We propose to interpolate the probability for the *bi*-gram composed by a verb+noun pair using an interpolation weight of

0.5 the higher- and lower-order *n*-grams (*i.e.*, we give the same weight for *bi*-grams and *one*-grams).

We may suffer from the same sparsity problem of language models when we consider the *bi*-gram composed of the verb+noun pair as the higher-order *n*-gram in our scenario[4]. Therefore, we experimented to interpolate the probabilities of our verb+noun pair (the high-order *n*-gram) with the probabilities of the pair's noun constituent (the low-order *n*-gram in this case if we follow the *n*-gram language model approach). Given this, the interpolated probability for the *bi*-gram composed by a verb+noun pair, after applying *Modified Kneser-Ney smoothing*, may be calculated as:

$$P_{interp}(v,n) \;=\; w \times [\alpha(n|v)] + (1-w)[\alpha(v)\gamma(n)] \tag{3.11}$$

where $\alpha(v)$ and $\gamma(n)$ are the functions involved on *Modified Kneser-Ney smoothing*; and w is the interpolation weight. The interpolation weight is the same for both sides of the sum (0.5). We stick with the same assumption regarding the pair's interpolated probability: if the interpolated probability of our verb+noun pair deviates too much from the interpolated probability of its similar variants in $S_{sim}(v,n) \cup \langle v,n \rangle$, the pair is lexically fixed and likely to be a VNIC. Therefore, lexical fixedness

[4]This is in fact true, given that we have pairs of words (verbs and nouns) or either verbs or nouns individually

based on interpolated probabilities is calculated as a standard z-score in a similar fashion to Equation 3.9.

3.2.4 Lexical Normalized Google Distance

So far, we have based our propositions on probabilistic and language models approaches. There are, however, other metrics that can be used to measure the association strength between two words. Thus, we also investigate the use of Normalized Google Distance (NGD) (Cilibrasi and Vitányi, 2007).

NGD is a metric that relies on page counts returned by a search engine on the Internet to measure the strength of the association between two words. For a pair of words x and y, NGD is defined as follows:

$$NGD(x,y) = \frac{max\{log(f(x)), log(f(y))\} - log(f(x,y))}{log(N) - min\{log(f(x)), log(f(y))\}} \quad (3.12)$$

where $f(x)$ is the frequency of the word x; $f(y)$ is the frequency of the noun y; $f(x,y)$ is the frequency of the word pair occurring together; and $f(*,*)$ is the count of all web pages in the search engine.

A property of NGD that makes it of interest for us is its smooth space of values, which is due to the removal of the dependency of multiplications in the formula. As we are interested in *VNIC type* identification in monolingual corpora, we decided to experiment with an NGD version that uses counts directly extracted from a corpus rather then returned

from a search engine on the Internet. Our NGD variant[5] is defined for a verb+noun pair v and n as:

$$NGD(v,n) = \frac{max\left\{\log(f(v)), \log(f(n))\right\} - \log(f(v,n))}{\log(f(*,*)) - min\left\{\log(f(v)), \log(f(n))\right\}} \quad (3.13)$$

where $f(v)$ is the frequency of the verb v occurring with any noun as its direct object; $f(n)$ is the frequency of the noun n occurring as a direct object of any transitive verb in the corpus; $f(v,n)$ is the frequency of the verb+noun pair occurring in a direct object relation; and $f(*,*)$ is the frequency of all verb+noun pairs in a direct object relation.

Lexical fixedness based on NGD also follows the assumption that if the NGD of the target verb+noun pair $\langle v,n \rangle$ deviates from the mean NGD of the set $S_{sim}(v,n) \cup \langle v,n \rangle$ the pair is likely to be a VNIC. Therefore, lexical fixedness based on NGD is calculated by computing a standard z-score following Equation 3.9.

At this point it is worth noting that although previous work has used counts obtained from the web in other types of multiword expression (MWE) identification, they did not focus on *idiom type* identification. For example, Keller and Lapata (2003) showed that the web can be used to obtain reliable counts for unseen bigrams in a corpus, and Ramisch et al. (2010) used the web as a corpus to obtain better counts for n-grams

[5]Where we also set $\log 0 = 0$, following *Information Theory* conventions, when any of the frequencies involved in NGD calculation is 0.

in a language model setting to identify English noun compounds.

3.2.5 Linear Fixedness Models

Similar to the baseline model, we can use each of these lexical fixedness metrics to compute an overall fixedness score for a verb+noun pair by combining each of them with the original syntactic fixedness metric using a weighted linear combination as in Equation 3.7. In the case of our proposed metrics, we replace the original baseline lexical fixedness with one of our own proposed metrics. Including the original baseline model described in Section 3.1.1.4, these combinations of syntactic and lexical metrics give us the following five models:

1. `Baseline`: Baseline syntactic + Baseline lexical
2. `Syntactic+Probabilities`: Baseline syntactic + Lexical based on probabilities (Lexical Probabilities)
3. `Syntactic+Smoothed`: Baseline syntactic + Lexical based on smoothed probabilities (Lexical Smoothed)
4. `Syntactic+Interpolated`: Baseline syntactic + Lexical based on interpolated back-off probabilities (Lexical Interpolated)
5. `Syntactic+NGD`: Baseline syntactic + Lexical based on NGD (Lexical NGD)

Throughout the rest of the Chapter we will refer to these models' by these names when we compare the performance of the fixedness metrics on different tasks.

3.3 VNICs Extraction Task

Having introduced a number of variants to overcome the limitations in PMI-based lexical fixedness and a set of alternatives to linear fixedness metrics (Section 3.2.5), in this section we present the evaluation of these models on a VNIC extraction task. This evaluation is concerned with investigating to what extent the fixedness metrics are useful in retrieving VNICs from corpora. We start by describing the data preparation (Section 3.3.1) followed by the presentation of the result of overall metrics (Section 3.3.2). We finish with our analysis of the metrics' performances comparing the *gain* of each approach (Section 3.3.3).

3.3.1 Data and Models Preparation

We parsed the written portion of the British National Corpus (BNC) (Burnard, 2007) using the Stanford CoreNLP parser (Manning et al., 2014). Subsequently we extracted all verb+noun pairs consisting of a transitive verb and a noun occurring in its direct object position as well the determiner introducing the noun. For every verb+noun pair that occurred in at least one of the syntactic patterns in Table 3.1, we recorded:

(a) the total count of that verb+noun pair occurring in any pattern, and (b) the total count of the verb+noun pair in each of the patterns.

Following these preparation steps, we applied Fazly's syntactic and lexical fixedness metrics as well as our own set of lexical fixedness versions to all recorded pairs given the counts obtained on the first step. To generate the similar verb+noun combinations we used the automatically built thesaurus of Lin (1998). As reported by Fazly et al. (2009), there is little variation in results for $20 \leq K \leq 100$ where $K = (K_v + K_n)$. We thus choose $K = 40$ (*i.e.*, $K_v = 20$ and $K_n = 20$) for all lexical models.

As outlined in Section 3.1.1.3, the overall fixedness measure has a parameter ω which needs to be set for the linear combination. As we are using the same corpus as Fazly et al. (2009) and taking their metrics as baseline for comparisons, we used the same value used by Fazly et al. ($\omega = 0.6$), as the most reasonable choice for all overall models[6].

After calculating all fixedness metrics we kept only those verb+noun combinations that occurs at least 10 times in the corpus (we did not take into account the determiners introducing the noun) following the same procedure as Fazly et al.. Indeed, we expect that this constraint will balance the distributions for all models tested. We then combined the metrics as defined in Section 3.2.5, after performing range normalization (range $[0, 1]$) on all new lexical fixedness metrics individually and on

[6]In fact we experimented with different values for ω but we found that the value reported by Fazly et al. is indeed the best choice for all the metrics. Thus, we do not report the combinations using other values of ω in this work.

Fazly's et al. syntactic fixedness[7].

3.3.2 Performance of Overall Metrics

To evaluate the retrieval performance, we selected the top 1,000 verb+-noun pairs ranked by each metric and checked in the Cambridge Idioms[8] and the Collins COBUILD[9] dictionaries whether the combinations were listed as idioms or not. If a verb+noun pair was listed as an idiom in at least one of these two dictionaries we considered it to be a VNIC. For each model we counted the number of VNICs that appeared in the top 1,000 verb+noun pairs as ranked by that model.

The results for the retrieval performance are presented in Table 3.2, divided in chunks of 200 verb+noun pairs. As we are evaluating the top 1,000 pairs ranked by each metric, we tested each rank produced by each metric against all the other ranks for significance using the Spearman's ranked correlation test (Spearman, 1907) and we found all $p \ll 0.05$.

Analyzing each of the subsets, we observed a high overlap across the top 1,000 verb+noun pairs defined by the different models: we found only 2,091 different pairs among the 5 lists. From those, a total of 414 verb+pairs were found to be VNICs, *i.e.*, around 21% of the top ranked verb+noun pairs have an idiom associated, with each model retrieving a different set of VNICs.

[7]Recall that Fazly's et al. lexical fixedness takes its values in the range $[0, 1]$.
[8]http://dictionary.cambridge.org/
[9]http://www.collinsdictionary.com/

Table 3.2
Results in terms of idioms found in the top 1,000 verb+noun pairs rank produced by each metric, in chunks of 200. We found only a total of 414 different VNICs in these subsets.

	Top				
Overall Metric	200	400	600	800	1,000
`Baseline`	35	79	127	177	238
`Syntactic+Probabilities`	**45**	**96**	**148**	**199**	**266**
`Syntactic+Smoothed`	45	78	107	151	203
`Syntactic+Interpolated`	33	67	100	143	190
`Syntactic+NGD`	21	48	84	119	170

3.3.3 Analysis of the Retrieval Task

As shown in Table 3.2, the performance of the metrics are not similar. The `Syntactic+NGD` metric had the worst result retrieving only 170 different VNICs among its top 1,000 verb+noun pairs list (*i.e.*, 17% of the verb+noun pairs were VNICs). We attribute this bad performance to the limitation imposed on the NGD metric by relying just on counts retrieved from the corpus[10].

The `Syntactic+Interpolated` gave the second worst result and found only 190 different VNICs (*i.e.*, 19% of the verb+noun pairs were VNICs). An analysis of different target verb+noun pairs revealed that most similar pairs listed in each set $S_{sim}(v, n)$, required to compute the final z-score for the lexical metric for each pair, do not occur in the corpus. Therefore, in those cases, the metric must rely on the counts of *one*-grams alone (which is the noun constituent). Consequently, we believe that the interpolation procedure is not sufficient to capture the degree of

[10]Recall that this metric were originally developed to consider counts retrieved from a search engine on the Internet.

deviation between the probabilities of the target verb+noun pair and its set of similar variants.

The third best result is from the *Syntactic+Smoothed* model, which found 20.3% (*i.e.*, 203 pairs) of VNICs among its top 1,000 verb+noun pairs. Although the smoothed version of probabilities produces a set $S_{sim}(v, n)$ with a more realistic set of probabilities for each verb+noun pair, it is still not sufficient to expose the correct difference between the target verb+noun pair probability and its similar variants.

The second best result is from the standard *Baseline*, which found 238 VNICs among its top 1,000 list (23.8%). As with the *Syntactic+Interpolated* metric, most of the similar pairs listed in $S_{sim}(v, n)$ do not occur in the corpus. Nevertheless, even with a sparse set of PMI scores, the metric captures a certain amount of the deviation between the target pair and its similar combinations which is then reflected on this metric producing the second best result.

The metric which produces most VNICs among its top 1,000 list is the *Syntactic+Probabilities* metric. Despite the fact that this metric also suffers with sparse scores for the pairs in the $S_{sim}(v, n)$, its approach of comparing the probabilities is able to capture the deviations in the z-score in a stronger way than the current state-of-the-art, producing 266 VNICs in its top list (26.6% of VNICs), an improvement of 2.8% over the baseline.

In total, the metrics combined found only 414 different VNICs in this

retrieval task. If we consider a situation where the idiom dictionary is being built from scratch and the cost budget is related to the number of verb+noun pairs that can be evaluated, we might want to know the number of verb+noun pairs that should be evaluated to retrieve all the VNICs related to that language. In other words, we might want to know which metric has the cheaper cost when used to gather the most number of VNICs using the lowest possible number of verb+noun pairs.

One way of interpreting this question is: how many pairs each metric must analyse to get all VNICs of that language? To answer this question, we simulate such a situation by considering the 414 VNICs found in the retrieval task the "universe" of VNICs in English. While in practice there is a much higher number of VNICs that exist in English, we consider this amount of VNICs to be sufficient to demonstrate the efficiency of each metric. With that in mind, we searched in the list of all verb+noun pairs ranked by each overall metric for the position of all the 414 VNICs. Table 3.3 shows the number of pairs that must be analysed by each metric to retrieve all the VNICs in the simulated "universe".

Looking at the results displayed in Table 3.3, we observe that *Syntactic+Probabilities* metric needs to analyse 6,000 verb+noun pairs less than Fazly' metric. Given the fact that we extracted 54,715 different verb+noun pairs from the corpus, *Syntactic+Probabilities* metric must analyse less than 20% of the corpus to get all VNICs in our simulated "universe", an advantage of 11.36% over the state-of-the-art and

Table 3.3
Number of verb+noun pairs that must be retrieved by each metric to get all 414 different VNICs found in the *idiom type* retrieval task and the correspondent percentage of the corpus.

Overall Metric	Number of verb+noun pairs	% of the corpus
Baseline	16,423	30.00
Syntactic+Probabilities	**10,203**	**18.64**
Syntactic+Interpolated	25,845	47.23
Syntactic+Smoothed	26,783	48.95
Syntactic+NGD	35,657	65.16

almost 30% over the third best result (*Syntactic+Smoothed*).

On Figure 3.1 we plot the *gain* for each metric regarding the retrieval task. The pairs are retrieved independently for each metric using its own rank of the verb+noun pairs.

As we can see from the figure that at 10% of the total number of verb+noun pairs, all metrics have found more than 80% of all VNICs. In addition, at the stage of 70% of all verb+noun pairs in the corpus, all 5 metrics have already found 100% of VNICs in the simulated "universe". At the very beginning, the 2 best metrics, *Syntactic+Probabilities* and *Baseline*, have a similar behaviour retrieving VNICs, with a slight advantage to *Baseline*, which finds more than 90% of the VNIC test "universe" before analysing 10% of the total number of verb+noun pairs in the corpus. Nevertheless, the *Syntactic+Probabilities* metric finds all the VNICs in the test "universe" before hitting 20% of the total number of verb+noun pairs, while *Baseline* finds all VNICs only after analysing 30% of verb+noun pairs.

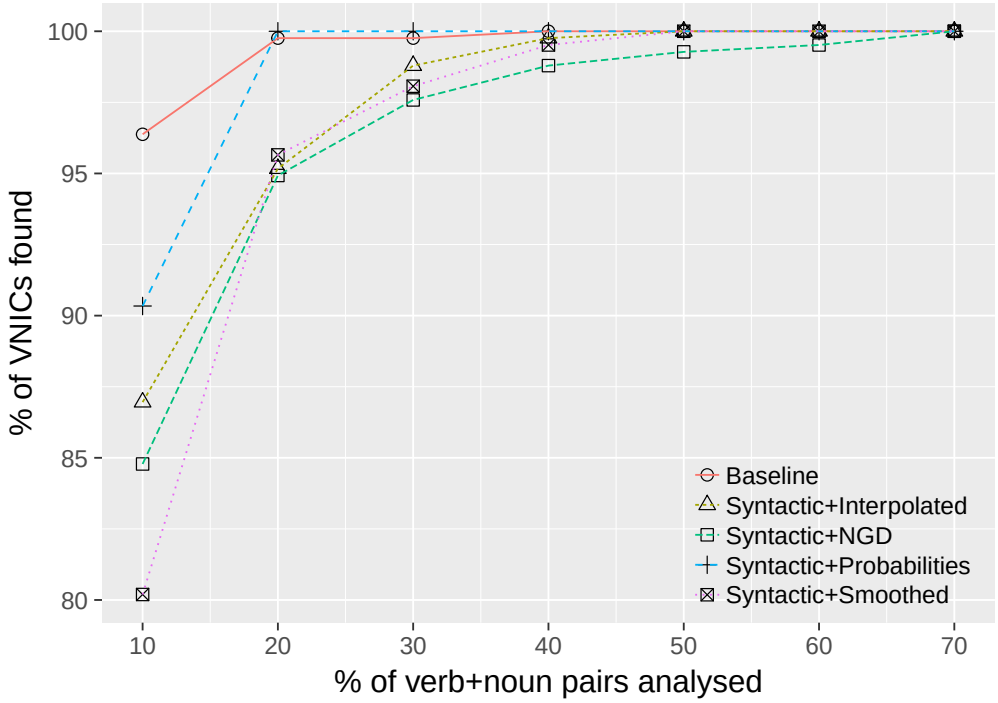


Figure 3.1
Gain chart for the overall metrics in the retrieval task. Note that with only 10% of verb+noun pairs analysed all metrics have found at least 80% of the VNICs in our in the simulated "universe".

The behaviour of `Syntactic+Probabilities` metric is advantageous when we consider a budget limit in retrieving VNICs from the corpora. Nevertheless, this is not the only way that fixedness metrics can be used to identify VNICs. In the next Section we provide an analysis of the fixedness metrics in a classification task, where the set of metrics must separate VNICs and non-VNICs in a test set.

3.4 VNICS Classification Task

Another application for fixedness metrics is to distinguish between VNICs and non-VNICs mixed in a set of verb+noun pairs. This situation

might happen when there are pairs already extracted from corpora but without any label assigned to those pairs. We would expect that, as VNICs will receive higher fixedness scores than literal verb+noun pairs, it is possible to label (*i.e.*, classify) VNICs and non-VNICs by setting a threshold on the fixedness score. Therefore, all pairs whose scores are higher than the threshold are labeled as VNICs.

Fazly et al. (2009) reported classification results over a balanced test set containing the same number of VNICs and literal pairs. Notably, in their work, Fazly et al. set the classification threshold to the median value of the test set, a procedure we believe is problematic. In this next section we perform an evaluation also using the median value as the threshold (Section 3.4.1) and compare the baseline performance (`Baseline`) to our own models (see Section 3.2.5). We discuss this issue further and provide a more suitable evaluation in Section 3.4.2

3.4.1 Baseline Evaluation

To make a fair comparison, we first replicate Fazly's et al. evaluation. We create a balanced test set by selecting VNICs and literal pairs (non-VNICs) from the 2,091 pairs found in the retrieval task (see Section 3.3.3). We constrained the selection process so that all of the selected pairs occurred with similar frequencies in the corpus, resulting in a set consisting of 95 VNICs and 95 literal pairs that we called "Test-Set".

In this classification task, we hope that all VNICs in the test set will

Table 3.4
Ordered results in terms of Precision, Recall and F1-Scores (calculated as a macro-average) for the classification task on a balanced dataset compared against a random baseline and setting the threshold to be the median score.

Model	Precision	Recall	F1-Score
Baseline	**0.89**	**0.89**	**0.89**
Syntactic+Probabilities	**0.89**	**0.89**	**0.89**
Syntactic+Smoothed	0.86	0.86	0.86
Syntactic+Interpolated	0.86	0.86	0.86
Syntactic+NGD	0.86	0.86	0.86

be ranked at the top whilst the literal verb+noun pairs will appear at the bottom of the list. In their work, Fazly et al. set the median score of the test set as the threshold for classification[11] and, to provide a direct comparison to their work, we also set the median score of each model as its threshold (we will come back to discuss a more suitable threshold in Section 3.4.2).

3.4.1.1 Baseline Results

Table 3.4 shows the classification task results in terms of Precision, Recall and F1-Score for all models (see Section 3.2.5). Given the fact that VNICs should have a higher fixedness score (assigned by the models) than literal pairs, we can say that the models are "ranking" the pairs. Therefore, we can test each "rank" against the others for significance using the Spearman's ranked correlation test (Spearman, 1907). We found all $p \ll 0.05$ (*i.e.*, each rank produced by a metric in the test set is statistical different than the ranks produced by the other metrics).

[11] As pointed out by Fazly et al. (2009), a suitable threshold should be determined based on development data

From Table 3.4 we see that in this particular setting most of the models have similar performance. Although *Syntactic+Interpolated*, *Syntactic+NGD* and *Syntactic+Smoothed* share the same Precision, Recall and F1-scores (0.86, 0.86 and 0.86 respectively), each model correctly classified a different set of verb+noun pairs. We attribute this strange behaviour to the median threshold used to classify the verb+noun pairs into VNICs and non-VNICs. We return to this issue of using the median as the classification threshold in the next section (Section 3.4.2).

The same trend is observed for the two best performing models: *Syntactic+Probabilities* and *Baseline*. Both share the same Precision, Recall and F1-scores (0.89, 0.89 and 0.89 respectively). These results indicates that each group of models should be capturing similar characteristics about the verb+noun pairs. These results are surprising given that, if the models are indeed capturing similar characteristics of VNICs, we would expect a similar set of performances in the retrieval task. In the next Section we analyse why the choice of threshold is biasing the evaluation and provide a more robust evaluation for the classification task.

3.4.2 Enhanced Evaluation

In the previous section we have shown that the models have similar performance when classifying VNICs and non-VNICs and that there are strong indications the evaluation were problematic given the choice of

thresholds. Thus, we decided to investigate a more relevant evaluation for the classification task.

Despite the fact that the studied models provide a real-valued measure of the fixedness of a given verb+noun pair, a thresholding function must be applied to construct a useful classifier. To do that, we need first to apply the logistic function to fixedness scores. The logistic function is defined as:

$$sigmoid(x) = \frac{1}{1 + \mathrm{e}^{-(x)}} \tag{3.14}$$

After this step, we apply the threshold to the fixedness scores obtained by each model. In the original work, Fazly et al. used the median value of their test set as the threshold. This procedure, however, undermines the evaluation over the balanced test set as it provides the model with information about the distribution of VNICs and non-VNICs in the test set. The major evidence for this behaviour can be observed comparing the results of retrieval task and the initial classification task. As the metrics had different performances when retrieving VNICs from the corpora, we expected the same differences to appear in the classification task. In other words, we expected that the models would not have the similar performances when classifying VNICs and non-VNICs as they had. Furthermore, even though Fazly et al. claim their model is "unsupervised", we cannot say that the approach is unsupervised in the "traditional" machine learning sense because of the need to set the threshold

for classification tasks. Even if we end up setting the median as the threshold, there is human effort to analyse which is the most suitable threshold. Using held-out data or manually looking into the test set for the median value, we are actually "supervising" the classifier by feeding it with information about the distribution of the data.

Framing the decisions about the threshold this way, we may consider the model as a supervised model rather than unsupervised. Therefore, we can use a more traditional approach to set the thresholds for each classification model. To do that, we first built a training set by selecting the remaining 319 VNICs held out from the 414 VNICs (found in the top 5,000 ranked verb+noun pairs by each metric) that we used to build the test set (see Section 3.4.1). To these VNICs, we added another 319 literal verb+noun pairs with similar frequencies sampled from the 1,582 remaining literal pairs[12]. We called this "Training-Set" and it contains 638 verb+noun pairs in total.

To decide which thresholds to apply, we performed a K-fold cross-validation (with $k = 3$) using the "Training-Set" to search for the best threshold for each model looking to maximize the F1-score in the training set. This step gave us 5 individual thresholds, one for each model, displayed in Table 3.5.

[12]Recall that due to a certain amount of overlap among the models, they retrieved only 2,091 different verb+noun pairs in their top 1,000 list and, from those, 414 pairs were found to be VNICs. From these 414, we sampled 95 VNICs and from the remaining 1,677 literal pairs we sampled another 95 pairs to include in the test set. This procedure left 1,582 remaining literal pairs which we then used to sample for the training set.

Table 3.5
Thresholds for each overall metric determined based on the F1-scores on the "Training Set" after applying the logistic function to the scores.

Model	Threshold
Baseline	0.63
Syntactic+Probabilities	0.64
Syntactic+Smoothed	0.61
Syntactic+Interpolated	0.62
Syntactic+NGD	0.62

We then proceeded by classifying the verb+noun pairs which scored equally or greater than the threshold as VNICs and, otherwise, as non-VNICs.

3.4.2.1 Performance on a Balanced Dataset

The results for each model in terms of Precision, Recall and F1-score are outlined in Table 3.6. All of the following results are statistically significant (by pairwise comparison among all models) according to McNemar's test (McNemar, 1947) (all $p \ll 0.05$).

From the analysis of Table 3.6 we can now observe a trend similar to the results of the retrieval task (Section 3.3.3). The worst performance is from the *Syntactic+NGD* model, which, as mentioned before, has its

Table 3.6
Ordered results in terms of Precision, Recall and F1-Scores (calculated as a macro-average) for the classification task on a balanced dataset.

Model	Precision	Recall	F1-Score
Baseline	0.83	0.75	0.74
Syntactic+Probabilities	0.82	0.73	0.70
Syntactic+Smoothed	**0.83**	**0.78**	**0.77**
Syntactic+Interpolated	0.79	0.64	0.59
Syntactic+NGD	0.78	0.62	0.56

classification power limited due to the fact that we constrained NGD to consider only counts obtained in the corpus. The second worst result, is from the *Syntactic+Interpolated* model, which is just slightly higher than the worst model. The intuition for the low results is that, when we apply the interpolation after smoothing the probabilities, we are actually reducing the difference between the probability of our target pair and the mean probability of the pair and its variants too much. In other words, we are "over-smoothing" the probability distributions across each target pair and its variants.

The *Syntactic+Probabilities* model has notably higher scores than the two worst models but, different from the retrieval task, it performs worse than the *Baseline*. This is a surprising result, given the fact that *Syntactic+Probabilities* metric had the best performance in both the retrieval and the baseline evaluation tasks. We attribute this to the fact that the *Syntactic+Probabilities* metric may also be assigning a high lexical fixedness to collocations when retrieving verb+noun pairs from a corpus. This behaviour might be a good characteristic when retrieving verb+noun pairs from a corpus but is not the best behaviour when classifying expressions when we do not know about their distribution. *Baseline* scored the second best among all fixedness models.

The best fixedness model is the *Syntactic+Smoothed* model. Analyzing this model one can also point out that, in a similar manner to the *Syntactic+Interpolated* model, the difference between the probability

of the target and the mean probability of its variants should be reduced and thus resulting in difficulties when classifying the verb+noun pairs. Nevertheless, we believe the higher results for this model are due to the fact that when we only smooth the probabilities and do not interpolate them, the deviations captured on the z-score are closer to the true deviations. We credit this to the application of *Modified Kneser-Ney smoothing* which is contributing to the metric being able to approach the true distribution of the verb+noun pairs, enabling us to capture the deviation of the probabilities of our target verb+noun pairs and their similar variants.

3.4.2.2 Performance on an Imbalanced Dataset

Although idioms do occur frequently in language, they do not occur as frequently as literal language. Consequently, evaluating VNIC type identification models on a balanced dataset may be misleading. In order to evaluate the models in a more plausible setting we created an imbalanced dataset that attempted to reflect the frequency of idiomatic usage in real language.

We used our results from the retrieval task as an estimate of the frequency of idioms in real language. There were on average 22% VNICs in the top 1,000 verb+noun pairs of each metric. To evaluate the models in a situation similar to this, we created an imbalanced test set to simulate a real corpus. To do that, we randomly selected 25 idioms out of the

Table 3.7
Results in terms of Precision, Recall and F1-Score (calculated as a macro-average) ordered by their F1-scores compared to *Baseline* in the classification task over an imbalanced test set.

Model	Precision	Recall	F1-Score
Baseline	1.00	0.44	0.61
Syntactic+Probabilities	1.00	0.44	0.61
Syntactic+Smoothed	**0.87**	**0.56**	**0.68**
Syntactic+Interpolated	1.00	0.28	0.43
Syntactic+NGD	1.00	0.21	0.35

95 identified for the balanced test set and mixed these with the 95 non-idioms also from the balanced test. We ran this classification 10 times (randomly sampling the 25 idioms each time) and we applied the classification models to the pairs on each run. To have a direct comparison with the results of the balanced test set we applied the same thresholds (see Section 3.4.2) on the imbalanced evaluation.

Table 3.7 present the results for running the classification task on the imbalanced test set in terms of Precision, Recall and F1-Score. The results reflect the average of each score after the 10 runs. Once again, all results reported are statistically significant (all $p \ll 0.05$), according to MacNemar's test.

As we can see in Table 3.7, the order of the results for the classification in this simulated corpus is the same as for the balanced test set. While the *Syntactic+NGD* model and the *Syntactic+Interpolated* model had very low F1-scores, the difference between *Baseline* and the *Syntactic+Smoothed* model is accentuated, rising from a difference of 0.03 to 0.08. An interesting point worth mentioning is the Precision

achieved for all models with the exception of the *Syntactic+Smoothed* model (the best performance model). All models had 1.00 Precision and low Recall scores (less then 0.5). At the same time, the *Syntactic+Smoothed* model had a lower Precision (0.87) than the other models and a higher Recall (0.56). If we consider these results together with those of the retrieval task, we can see that the *Syntactic+Smoothed* model is more "conservative" than *Baseline* and the *Syntactic+Probabilities* model (the 2 best performances on the retrieval task, see Table 3.2) and thus, even with lower Precision, it has a better Recall reducing the number of false positives in the final result (*i.e.*, classifying literal verb+noun pairs as VNICs).

3.5 Kernel-Based Models

In the previous section we presented a classification task evaluation of models built directly from our fixedness metrics. While these results were good and demonstrated a strong improvement over the state-of-the-art, we believed that further improvements were achievable. In our investigations, we identified a high non-linearity in the decision boundary between VNICs and non-VNICs based on the plots of the scores returned by the five fixedness models as displayed in Figure 3.2.

One of the limitations of the fixedness models for classifying VNICs is the weighted linear combination used to merge the syntactic and lex-

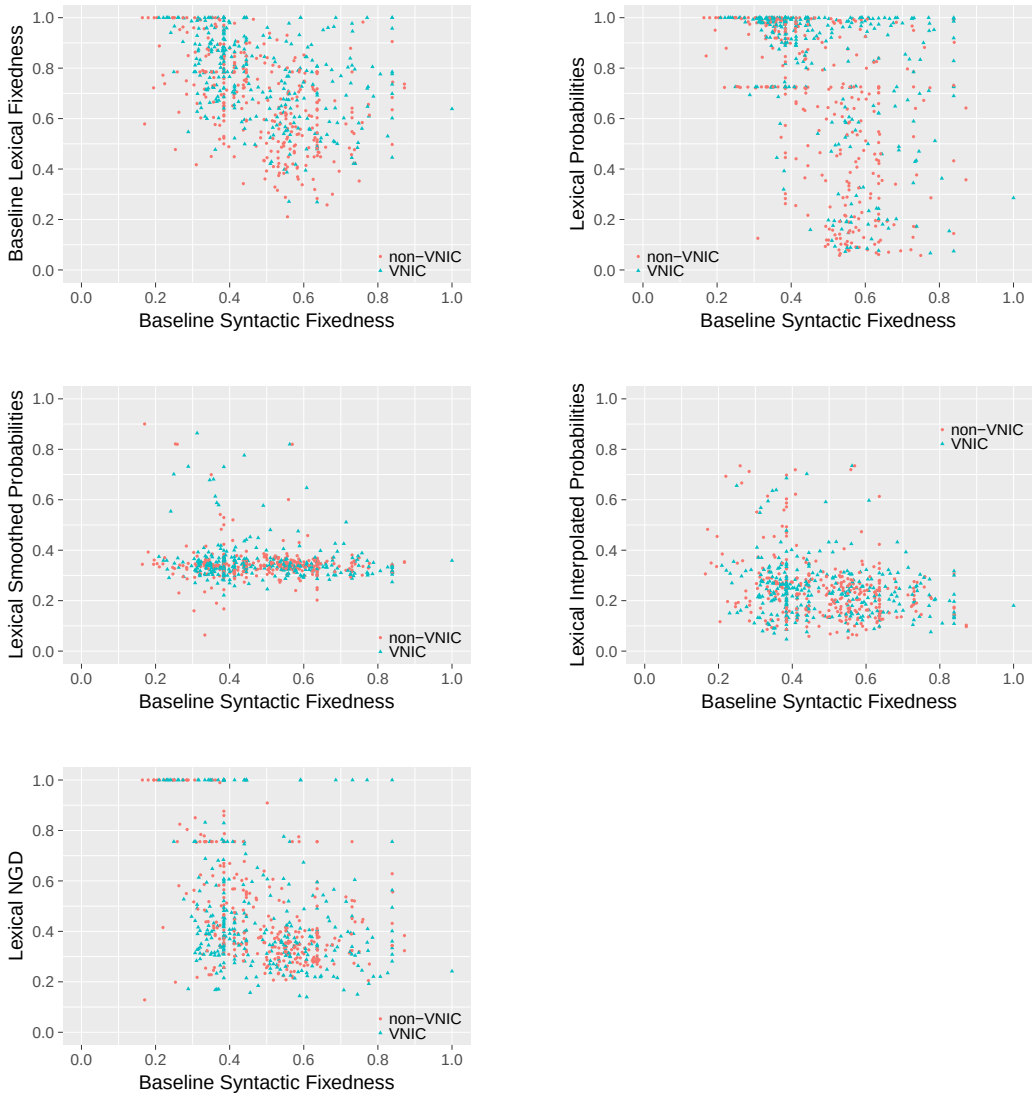


Figure 3.2
Distribution of VNICs and non-VNICS scored by Syntactic and Lexical fixedness metrics. Note that due to the use of a small scale for the images, some verb+noun pairs with very similar values for one or the other metric seems to have the same value and form a "line" in the plots.

ical metrics, which is not able to model the non-linear decision boundaries. Therefore, we propose to use the fixedness metrics as features for a classifier capable of modeling non-linear decision boundaries. For our set of experiments, we selected the Support Vector Machine (Vapnik, 1995) as they can be quickly trained and are less prone to overfitting

than other similar models (Kelleher et al., 2015).

In the following sections we give a brief description of the chosen SVM (Section 3.5.1), we then describe how we train the model by using the previously introduced fixedness metrics as features (Section 3.5.2) and we provide the results for a classification task on both balanced (Section 3.5.3) and imbalanced test sets (Section 3.5.4).

3.5.1 Soft-Margin Support Vector Machines

The SVM is a classification tool designed to find the optimal hyperplane that maximizes the distance between two classes (Zaki and Meira Jr., 2014). An SVM projects the input features into a higher-dimensional feature space and attempts to find a linear separating hyperplane in this higher-dimensional space. The intuition is that a linear separating hyperplane may exist in the higher-dimensional feature space even though the classes are not linearly separable in the original input feature space[13] (Kelleher et al., 2015). For the cases where the classes are not perfectly linearly-separable even in the higher-dimensional feature space the SVM introduces "slack variables" for each datapoint which indicates how much that point violates the separable hyperplane. Then, the goal of the SVM training turns into finding the hyperplane with the maximum margin that also minimizes the slack terms. This SVM structure is called a "Soft-margin SVM".

[13] An thus, generating a non-linear boundary on the original input feature space.

The task of training an SVM with a *linear kernel* is usually framed as a constrained quadratic programming problem but, in its native form, it is an unconstrained empirical loss minimization including a penalty term for the classifier being learned in direct space (Shalev-Shwartz et al., 2007). Framed this way, the *linear kernel* SVM can be trained by solving a loss minimization problem applying Stochastic Gradient Descent (SGD) (Bottou, 2010).

3.5.2 Building SVM Models from Fixedness Metrics

Based on the properties of SVMs, we decided to experiment with a Soft-margin SVM with a *linear kernel* using the fixedness metrics as input features[14]. Recall that, even with a *linear kernel*, the features that are non-linearly separable in input space (where all fixedness metrics operate) might become linearly separable in feature space (where the SVM will operate) given the transformations performed by the "linear kernel" to the features.

We trained the model with SGD, following Bottou (2010). To train a model in such a configuration, a parameter α (a constant that multiplies the regularization term) and the type of regularization function must be set. To achieve the best configuration for the model, we performed a grid search using k-fold cross-validation ($k = 3$) over the "Training-Set"

[14]We also experiment with other kernel functions but the scores were lower than the linear models. Therefore we do not report them in this thesis.

(see Section 3.4.2) using all individual metrics[15] as features. Based on the cross-validation, we set $\alpha = 0.0001$ and the regularization function to be the L2-norm. This step gave us a 6-feature SVM which we called SVM-All. We trained it for 20 epochs (*i.e.*, 20 iterations over the entire training set).

Given the different performance of the metrics in both retrieval and classification tasks (Sections 3.3.3 and 3.4.2.2 respectively), we decided to perform feature selection before training a second SVM. We follow Guyon et al. (2002) and select the "best features" using the values of the weights of an already trained SVM, which, in this case, is SVM-All. We selected the three features with the higher weights: Baseline syntactic fixedness; Baseline lexical fixedness; and the lexical probabilities[16]. We performed a grid search for the best parameters to fit this SVM with these three features using k-fold cross-validation ($k = 3$) over the "Training-Set". Based on the results, we set $\alpha = 0.01$ and set the regularization function to be the L1-norm. This step gave us a 3-feature model which we called SVM-Select. We trained this model for 20 epochs.

As a matter of comparison, we trained a third SVM model using only the Baseline metrics as features. Again, we performed a grid search using k-fold cross validation ($k = 3$) over the "Training-Set" to set the

[15]Baseline syntactic, Baseline lexical, Lexical probabilities, Lexical smoothed, Lexical Interpolated and Lexical NGD

[16]These 3 features had notably higher weights in the final decision function than the other features: Baseline syntactic: 0.64819684; Baseline lexical: 1.46668879; Lexical Probabilities: 1.93554506; Lexical Smoothed: 0.22613072; Lexical Interpolated: -0.28795421; and Lexical NGD: -0.8517920.

Table 3.8
Results in terms of Precision, Recall and F1-Score (calculated as a macro-average) of the 3 SVM models compared to our 2-best linear models: *Baseline* and *Syntactic+Smoothed*.

Model	Precision	Recall	F1-Score
SVM-All	0.80	0.78	0.78
SVM-Select	**0.87**	**0.85**	**0.85**
SVM-Baseline	0.83	0.73	0.71
Syntactic+Smoothed	0.83	0.78	0.77
Baseline	0.83	0.75	0.74

α parameter and the regularization function ($\alpha = 0.0001$ and L2-norm respectively). We call it SVM-Baseline and we trained it for 20 epochs.

3.5.3 Classification on a Balanced Test Set

In Table 3.8 we present the results of the three SVM models compared to the 2 best linear models: *Baseline* model and the *Syntactic+Smoothed* model. All the results are statistically significant according to McNemar's test (all $p < 0.05$).

As we can see in Table 3.8, both "SVM-All" and "SVM-Select" outperformed the linear fixedness models in the classification task on a balanced dataset. The "SVM-Baseline" performed worst among all 5 models compared in this section. The results obtained by the "SVM-All" model are, to our surprise, only slightly higher than those obtained by the best linear models (*Syntactic+Smoothed* model and *Baseline* model). Our intuition is that some of the features are less relevant than others and, thus, impacting negatively on the final classification result. The best results in this scenario is from the "SVM-Select", with the reduced

number of features.

A final point worth considering is the type of errors each of the models is prone to make. Taking the VNIC class as the positive class, most of the errors for the three SVM models and our *Syntactic+Smoothed* model models were false negatives (they classified VNICs as non-VNICs). By comparison, the other models (*Baseline* and our other models) all had higher rates of false positives. In our opinion, within this context, false positives are more problematic than false negatives given that a false positive would result in a non-idiom being included in an idiom dictionary.

3.5.4 Classification on an Imbalanced Test Set

As with the case of the linear models, evaluating an *idiom type* classifier over a balanced test set might be misleading. Thus, we repeated the process of building an imbalanced test set following the procedure adopted in Section 3.4.2.2 and used the same SVMs developed in the previous section. We also ran this evaluation 10 times and report the averaged results.

In Table 3.9 we present the results in terms of Precision, Recall and F1-Score of the three SVM models compared to the 2 best linear models on the imbalanced test set: *Baseline* model and the *Syntactic+Smoothed* model. We found that all the results are statistically significant according to McNemar's test (all $p < 0.05$).

Table 3.9
Results in terms of Precision, Recall and F1-Score (calculated as a macro-average) of the 3 SVM models compared to our 2-best linear models, *Baseline* and *Syntactic+Smoothed* models, on the imbalanced test set. The results are ordered by their F1-scores.

Model	Precision	Recall	F1-Score
SVM-All	**0.76**	**0.76**	**0.76**
SVM-Select	0.72	0.60	0.65
SVM-Baseline	0.48	1.00	0.64
Syntactic+Smoothed	0.87	0.56	0.68
Baseline	1.00	0.44	0.60

Analysis of Table 3.9 shows a different result than the SVM classification in a balanced test set: the "SVM-All" is now the model with the best performance followed by a linear model (*Syntactic+Smoothed* model). The best model in the balanced test set, "SVM-Select" is now only the third best result, 0.11 F1-Score points behind "SVM-All". This scenario aligns with the results of linear models in Section 3.4.2.2, where we found the model with Lexical probabilities to have a worse performance when compared to the best models and, thus, is not so suitable for a classification in an imbalanced test set. In addition, the results are indications that the other metrics are contributing to smooth the process of classifying VNICs and non-VNICs with "SVM-All" in its higher-dimensional feature space, allowing this model to perform well in the imbalanced test set.

3.6 Discussion

In this chapter we presented a set of new lexical fixedness metrics and evaluated how they can be used in a linear classifier for idiom type identification. We also proposed the the use of a non-linear SVM classifier for idiom type identification and demonstrated how this model can improve on the current state-of-the art in idiom type identification. However, despite this progress there is still much more work needed to progress idiom type identification. In particular, many of the fixedness metrics struggle when applied to small corpora and, hence, idiom type identification is still an open challenge for under-resourced languages. Indeed, the work presented in this chapter focused on idioms in English: English is a well-resourced language with English idiom dictionaries easily available online (for a fee) and this enabled us to create the datasets required for our work. Doing this type of work would be much more challenging in an under-resourced language.

Another perspective that can be taken on the research presented in this chapter is that idiom type identification is challenging even when linguistic resources are available. The fixedness metrics that were the basis of this work struggled to capture the idiosyncratic nature of idioms. For example the probabilistic approaches with smoothing techniques, which have presented good results in practice for language modelling, do not capture entirely the degree of fixedness presented by idioms. This

situation is clear when we analyse the verb+noun pairs ranked by each metric. The VNICs, which were supposed to have higher degrees of both syntactic and lexical fixedness, were mixed between literal verb+-noun pairs, constituting the minority of the pairs at the top of the rank. This is an undesirable situation for VNIC type identification. Moreover, the previous probabilistic approaches, which were inspiration for our work, were developed to capture information carried on pairs of words that should be literal usages of language, where the words are mainly used with their regular meaning (*i.e.*, the meaning recorded in the lexicon). The use of such techniques to capture "idiomatic information" between pairs of words might fail given the idiosyncratic behaviour of such expressions.

The other approaches to measuring fixedness that we have explored such as PMI and NGD also have flaws on the VNIC type identification task. While PMI, given its properties, is difficult to understand when we consider the similar verb+noun pairs, we would expect that a metric with a smoother space of values, such as NGD, would present us with some information regarding idiomatic pairs, this however was not the behaviour observed in our experiments. Nevertheless, we are aware that we have used a modified version of NGD which considers counts obtained only in the corpus instead of the original version that considers counts from a search engine on the internet – in other words a much larger corpus. Notwithstanding the limitations of applying NGD to a re-

stricted corpus we believe that overall the results of our fixedness based experiments (be they based on PMI, NGD or probability based measures) indicates that more sophisticated representations, that are in turn able to model more nuanced and complex interactions, are necessary for idiom processing.

As an initial move towards more sophisticated representations we used an SVM trained on the fixedness metrics as a non-linear classifier for the task. The move to using a non-linear classifier was motivated by the non-linear boundary between VNICs and non-VNICs that was evident in the scatterplots of the corpus in the fixedness metrics feature space.

The use of the SVM model improved on the current state-of-the-art in idiom type identification. We believe that this improvement was due to the fact that an SVM projects the input features into a different space that includes interaction terms between these features and that modeling these interactions permits the SVM to accommodate some of the idiosyncratic aspects of idioms. We will return to the need for more complex representations of idioms throughout this thesis.

3.7 Conclusions

In this chapter we analysed the state-of-the-art in idiom type classification. We pointed out their limitations and proposed a new set of lexical

fixedness metrics to overcome the problems arising from using Pointwise Mutual Information (PMI) as the basis for the metric. We also showed some flaws in the original evaluation of the fixedness metrics and described 2 different tasks to evaluate the metrics: a retrieval and a classification task, where the latter were further split into classifying VNICs over a balanced test set (a direct comparison with previous work) and over an imbalanced test set (which provides a more realistic simulation of the actual distribution of VNICs in English).

We showed that a probabilistic approach (such as the *Syntactic+Probabilities* model) is most suited for retrieving VNICs types from corpora. We demonstrated the "gain" obtained by using such a model to retrieve a simulated "universe" of VNICs from all verb+noun pairs in a real corpus, where this approach finds all VNICs types after analyzing a lower number of verb+noun pairs than the current state-of-the-art.

We also presented a direct comparison with the evaluation method used in previous work and pointed out the problems with that method, *i.e.*, where the use of a median threshold permitted the model to indirectly glean information about the class distribution in a balanced test set, and, furthermore, the fact that a balanced (idiom versus non-idiom) test set is an unrealistic simulation of the real distribution of idioms in language. We provided a more suitable evaluation using a held-out training set to search for the models' best parameters and also created a test corpus with a more realistic idioms versus non-idioms distribution. In

this evaluation we showed that one of our new proposed lexical metrics (*Syntactic+Smoothed* model) out-performs the state-of-the art on a classification task over both a balanced test set (to provide a direct comparison to previous work) and our imbalanced test set (which we believe provides a more plausible estimate of the generalization power of models in language).

We also experimented to feed the fixedness metrics as features to SVM and train this model to classify verb+noun pairs. The most important point to note is that the classification with the SVM revealed that even in a task where the discourse history can be ignored, the alternative projection of features representing idioms improves the performance of the classifiers due to the interactions between features. In the next chapter we study the use of distributed semantics representations to model sentences for the task of idiom token identification.

Chapter 4

Idiom token Identification

A strategy often adopted by SMT and other NLP systems for processing idioms is to use a pre-compiled idiom dictionary as part of the system and then to use the information contained in the dictionary to process an idiom correctly. For this strategy to work it is useful for an NLP system to be able to identify the presence of an idiom in its input. Indeed, not only must a system be able to identify that a pattern of words matching an idiom in a dictionary is present in the input it must also be able to identify that the intended meaning of these words is the idiomatic rather the literal sense. In earlier work, we demonstrated that idiom token identification (or classification) can usefully be used as a pre-processing step to inform an NLP system about the presence of an idiom in an input and thereby enable the system to process the idiom correctly (Salton et al., 2014b). In this chapter we address the task of *idiom token* identification, which is often framed as a problem in terms of modeling the global lexical context.

As we will see, previous work in *idiom token* classification tried to capture the fact that an idiomatic expression, *e.g. break the ice*, is likely to have a literal meaning in a context containing words such as *cold*, *frozen* or *water* and an idiomatic meaning in a context containing words such as *meet* or *discuss* (Li and Sporleder, 2010a). In addition to that, a number of papers on both *idiom token* and *idiom type* identification have shown that a range of other features could also be useful for idiom token classification, including local syntactic and lexical patterns (Fazly et al., 2009). A potential problem, however, with these approaches is that, in general, these non-global features are specific to a particular phrase and, as a result, these models create a different classifier for each idiomatic phrase. A key challenge on training classifiers in such a setting, is to identify from a set of useful features which are the correct ones to use in *idiom token* classification for a specific expression turning this approach into a expensive and difficult to scale task.

Humans, however, process idioms efficiently. Currently, one of the most accepted psycholinguistic theories for human idiom recognition is the "Configuration Hypothesis", proposed by Cacciari and Tabossi (1988). This theory assumes a single cognitive lexicon but two different pathways for processing idioms and literal language. The theory posits that humans start processing all words in a stream in the same way until they find an "idiomatic key", which may be a word or a combination of words that may or may not be part of an idiom. Indeed, what functions

as an idiomatic key can vary from one expression to another and can even vary for a given expression from one context to the next. When the "idiomatic key" is found, the human brain deviates the process from the "regular pathway" of language processing and activates a different region of the brain which then deals with the figurative content of the stream. Therefore, idioms are not stored as lexical items as initially assumed by other theories of human idiom recognition and, therefore, they would not be stored in a "separate lexicon" in the human memory (Cacciari and Tabossi, 1988).

One of the main implications of the "Configuration Hypothesis" is that, given the fact that idioms are processed as any sequence of words until the figurative meaning is taken, their constituent words must have polysemous meanings and the human brain must select the appropriate meaning for these words *on-line*. As a consequence, this behaviour is only possible if words are mapped into distributed representations into the brain (Vega-Moreno, 2001). Moreover, neurocognitive studies have found evidence supporting the fact that distributed representation of word concepts and parallel processing of literal and figurative language in human brain does occur (Rommers et al., 2013).

Meanwhile, there has been an explosion in the use of neural networks for learning "machine readable" distributed representations (or distributed semantics) for NLP (e.g., Socher et al. (2013), Kalchbrenner et al. (2014) and Kim (2014)) in recent years. These types of representations

are automatically learned from a corpus and are able to encode multiple linguistic features simultaneously. For example, word embeddings can encode gender distinctions and plural-singular distinctions (Mikolov et al., 2013) and the representations generated in sequence-to-sequence mappings have been shown to be sensitive to word order (Sutskever et al., 2014). Moreover, the development of techniques such as Skip-Thought Vectors (or *Sent2vec*) (Kiros et al., 2015) have provided an approach to learn distributed representations of sentences in an unsupervised manner.

Inspired by these findings in neurocognitive studies and the recent interest in distributed representations[1] in NLP, we explore the question of whether the representations generated by *Sent2vec* encodes features that are useful for *idiom token* classification. This question is of particular interest given the fact that *Sent2vec* models take only the sentence containing the candidate expression as its input whereas the baselines systems use paragraphs surrounding that sentence. In addition, we further investigate the construction of a "general" classifier that can make predictions irrespective of the candidate expression, using just the distributed representation generated for that sentence. This approach is in stark contrast with previous work on *idiom token* classification that has primarily adopted a "per-expression" classifier approach. This approach generates a different classifier for each expression and has been based on

[1]From now on, we will refer to "machine readable" distributed representations as distributed representations. We will make explicit distinction when referring to representations in the human brain.

a range of more elaborated context features, such as discourse and lexical cohesion between the sentence and the larger context. We show that our method achieves competitive performance compared to the state-of-the-art methods using far less contextual information, making it an important contribution to a range of applications that do not have access to a full discourse context.

We begin our investigations by reviewing previous work on *idiom token* identification in Section 4.1, including details of the best performing model for *idiom token* identification, the TopSpace algorithm of Peng et al. (2014), which we use as our baseline. We outline the details of the Skip-Thought Vectors model of Kiros et al. (2015) that we use to extract the distributed semantics for our experiments in Section 4.2. In Section 4.3 we delineate our experiments on both "per-expression" and "general" settings and we discuss the challenges faced when running these experiments. We present our results in Section 4.4 with an extend the discussion in Section 4.5. Finally we draw our conclusions in Section 4.6.

4.1 Previous Work on *Idiom token* Identification

A complicating factor about idioms for NLP systems is the fact that idioms can share surface realizations with literal usages of their constituents. To overcome the problems posed by this property of idioms,

idiom token identification is applied to distinguish between idiomatic and literal usages of potentially idiomatic phrases (Fazly et al., 2009).

One of the earliest works on *idiom token* classification was on Japanese idioms (Hashimoto and Kawahara, 2008). This work used a set of features, similar to those used in Word Sense Disambiguation (WSD) research, that were defined over the text surrounding a phrase, as well as a number of idiom specific features. These features were in turn used to train an SVM classifier based on a corpus of sentences tagged as either containing an idiomatic usage or a literal usage of a phrase. Their results indicated that the WSD features worked well on *idiom token* classification but that their idiom specific features did not help on the task for some particular expressions. As the authors report, some of the idiomatic expressions were very infrequent in the dataset and the task of finding them could be more difficult for the classifier even with idiom specific features.

Fazly et al. (2009) developed the concept of a canonical form (defined in terms of local syntactic and lexical patterns) while focusing on *idiom token* classification in English. In their work the authors argued that each idiom has a distinct canonical form (or a small set of canonical forms) that distinguishes the idiomatic usages of a phrase from its literal usages. Around the same time, a model based on the strength of the link between the structure of the discourse and the candidate expression was proposed by Sporleder and Li (2009). In that model, a weak link

between the expression and the discourse indicates an idiomatic phrase.

In related work, Li and Sporleder (2010a) experimented with a range of features for *idiom token* classification models, including: global lexical context, discourse cohesion, syntactic structures based on dependency parsing, and local lexical features such as cue words, occurring just before or after a phrase. An example of a local lexical feature is when the word *between* occurs directly after *break the ice*; here this could mark an idiomatic usage of the phrase: *it helped to break the ice between Joe and Olivia*. The results of this work indicated that features based on global lexical context and discourse cohesion were the best features to use for *idiom token* classification. The inclusion of syntactic structures in the feature set provided a boost to the performance of the model trained on global lexical context and discourse cohesion.

Taking a slightly different approach, Li and Sporleder (2010b) proposed a model based on the assumption that figurative and literal language are generated by two different Gaussians distributions. The representation of the model is based on the semantic relatedness similar to those used by Sporleder and Li (2009). The authors employ the Expectation Maximization method to train a Gaussian Mixture Model to classify the candidate expression. The classification is performed by choosing the category that maximizes the probability of fitting one of the Gaussian components. The results of Li and Sporleder (2010b) confirmed that figurative language does not present the same cohesion level

with the surrounding context as literal language.

Work on *idiom token* classification often frame the problem in terms of modelling the global lexical context, similar to Sporleder and Li (2009). For example, these models try to capture the fact that the idiomatic expression *break the ice* is likely to have a literal meaning in a context containing words such as *cold*, *frozen* or *water* and an idiomatic meaning in a context containing words such as *meet* or *discuss* (Li and Sporleder, 2010a). Frequently these global lexical models create a different *idiom token* classifier for each idiomatic phrase. However, a number of papers on *idiom type* and token classification have pointed to a range of other features that could be useful for *idiom token* classification, including: local syntactic and lexical patterns (Fazly et al., 2009); and cue words (Li and Sporleder, 2010a). However, in most cases these non-global features are specific to a particular phrase. So a key challenge is to identify from a range of features which features are the correct features to use for *idiom token* classification for a specific expression.

Following this "per-expression" approach, Feldman and Peng (2013) framed their work on *idiom token* classification as a problem of outlier detection. The intuition of their work is that given the fact that idiomatic usage will exhibit a weak cohesion with its surrounding context, idiomatic usages will be semantically distant from the topics of the discourses in which they are present. Therefore, a candidate expression is likely to be an idiom if it is a semantic outlier with respect to the sur-

rounding context. The authors explore two different approaches to the outliers detection that are based on principal component analysis (PCA) and linear discriminant analysis (LDA) respectively.

Building on this work, Peng et al. (2014) explore the assumption that candidate expressions contained in a given segment (*e.g.*, a paragraph) that are semantically similar to the main topic of that segment are likely to be literal usages. They extract topics to build their representation for each segment within a corpus. Given the extracted representations, the authors classify a candidate expression as a literal or idiomatic usage by projecting it into a topic space representation, using the extracted topics, and labeling the outliers in that space as idioms. We present the model of Peng et al. (2014) (our baseline) in more details in Section 4.1.1.

Interestingly, unlike the majority of previous work on *idiom token* classification, (Li and Sporleder, 2010a) also investigated building general models that could work across multiple expressions. Again they found that global lexical context and discourse cohesion were the best features in their experiments. Although their general models achieved good results the performance was still below that of classifiers designed and trained for individual expressions.

4.1.1 Peng's TopSpace Algorithm

Peng et al. (2014) described a method suited to *idiom token* classification based on topic modeling. The authors explore the assumption that

candidate expressions contained in a given segment (*e.g.*, a paragraph) that are semantically similar to the main topic of that segment are likely to be literal usages.

Their algorithm, called *TopSpace*, first selects single or multi paragraph segments from a corpus, where each segment is centered at the paragraph containing either an idiomatic or literal usage of a particular expression. Once the segments are defined, the algorithm applies Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to extract the topics that best represent each segment. For each of the segments within the corpus, the algorithm organizes the extracted topics into a "topic-term document matrix" ($M_{\hat{D}}$), where terms are topic terms and documents are segment topics.

Once $M_{\hat{D}}$ is extracted, the algorithm builds another "topic-term document matrix" (M_Q) to the document containing the candidate expression to be classified. Once $M_{\hat{D}}$ and M_Q are extracted, the algorithm applies Fisher's Linear Discriminant Analysis (FDA) (Fukunaga, 1990) to $M_{\hat{D}}$ and projects M_Q into that space. Based on these lower-dimensional representations, the authors trained a number of classifiers to label the candidate expression as idiomatic if M_Q, when projected into the lower-dimensional space, is an outlier to the $M_{\hat{D}}$ topics.

Although the work of Peng et al. (2014) achieves state-of-the-art results in *idiom token* classification, it has it has several disadvantages. For example, the best performing version of the algorithm breaks the corpus

into segments of five paragraphs: two paragraphs before the paragraph containing the candidate expression; the paragraph that contains the candidate expression; and two paragraphs after the paragraph containing the candidate expression. Also, to perform the classification, the algorithm expects the same amount of context from the document containing the candidate expression. However, that amount of surrounding context is not always available at the input. Therefore, to be useful as a component in many NLP systems, a method for *idiom token* classification must be able to extract the relevant features using smaller amounts of input content.

4.2 Skip-Thought Vectors

Skip-Thought Vectors (or *Sent2vec*) (Kiros et al., 2015) is an example of a distributed semantics model for NLP. Skip-Thought Vectors is an application of the Encoder/Decoder architecture (Sutskever et al., 2014), based on Recurrent Neural Networks (RNN), originally designed for Neural Machine Translation (NMT) (Bahdanau et al., 2015): the encoder receives an input sentence and maps it into a distributed representation (*i.e.*, a fixed-length vector of real numbers); and the *Sent2vec* decoder, which is essentially a language model conditioned on that representation, is used to "predict" the sentences surrounding the input. As a consequence of this behavior, the encoder learns, amongst other things,

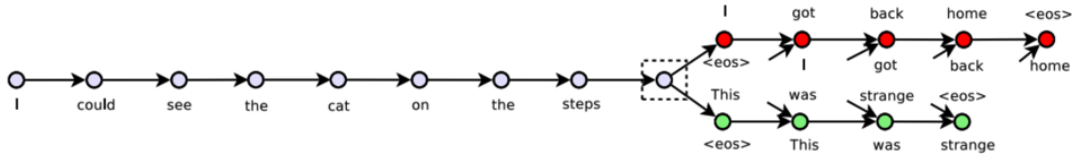


Figure 4.1
Picture representing the Encoder/Decoder architecture used in the Sent2Vec as shown in Kiros et al. (2015). The gray circles represent the Encoder unfolded in time, the red and the green circles represent the Decoder for the previous and the next sentences respectively also unfolded in time. In this example, the input sentence presented to the Encoder is *I could see the cat on the steps*. The previous sentence is *I got back home* and the next sentence is *This was strange*. Unattached arrows are connected to the encoder output (which is the last gray circle).

to encode information about the context of the input sentence without explicitly accessing that context, based solely on that input. Figure 4.1 shows the architecture of *Sent2vec* as presented by Kiros et al. (2015).

More formally, let the tuple (s_i, s_{i-1}, s_{i+1}) represent, respectively, the input sentence, the previous sentence to the input and the next sentence to the input. Let w_i^t denote the *t*-th word of s_i and $\mathbf{x}_i^t$ denote its word embedding. Following Kiros et al. (2015), we describe the model in its three constituent parts: the encoder; the decoder; and the objective function.

Encoder. Given the sentence s_i of length N, let $w_i^1, \ldots, w_i^N$ denote the ordered sequence of words in s_i. At each timestep t, the encoder (a RNN with Gated Recurrent Units - GRUs (Cho et al., 2014)) produces a hidden state $\mathbf{h}_i^t$ that represents the sequence from w_i^1 up to w_i^t. Therefore, $\mathbf{h}_i^N$ represents the full sentence. Each $\mathbf{h}_i^N$ is produced by iterating the following equations (without the subscript i):

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^e \mathbf{x}^t + \mathbf{U}_r^e \mathbf{h}^{t-1}) \tag{4.1}$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z^e \mathbf{x}^t + \mathbf{U}_z^e \mathbf{h}^{t-1}) \tag{4.2}$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^e \mathbf{x}^t + \mathbf{U}^{\mathbf{e}}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \tag{4.3}$$

$$\mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \tag{4.4}$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product.

Decoder. In *Sent2vec*, two RNNs are used as decoders (one for the sentence s_{i-1} and the other for the sentence s_{i+1}) with different parameters except the embedding matrix ($\mathbf{E}$). Each decoder is a neural language model conditioned on the distributed representation $\mathbf{h}_i^N$, generated for the input sentence s_i. A new set of matrices ($\mathbf{C}_r$, $\mathbf{C}_z$ and $\mathbf{C}$) are introduced in each of the decoders to condition the GRU on $\mathbf{h}_i^N$. Let $\mathbf{h}_{i+1}^t$ denote the hidden state of the decoder of the sentence s_{i+1} at time t. Decoding s_{i+1} requires iterating the following equations (without the subscript $i + 1$):

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^d \mathbf{x}^t + \mathbf{U}_r^d \mathbf{h}^{t-1} + \mathbf{C}_r \mathbf{h}_i^N) \tag{4.5}$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z^d \mathbf{x}^t + \mathbf{U}_z^d \mathbf{h}^{t-1} + \mathbf{C}_z \mathbf{h}_i^N) \tag{4.6}$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^d \mathbf{x}^t + \mathbf{U}^{\mathbf{d}}(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{C}\mathbf{h}_i^N) \tag{4.7}$$

$$\mathbf{h}_{i+1}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \tag{4.8}$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product. An analogous computation is required to decode s_{i-1}.

Given $\mathbf{h}_{i+1}^t$, the probability of the word w_{i+1}^t conditioned on the previous $w_{i+1}^{<t}$ words and the encoded representation produced by the encoder (h_i^N) is:

$$P(w_{i+1}^t | w_{i+1}^{<t}, h_i^N) \propto exp(\mathbf{E}_{w_{i+1}^t} \mathbf{h}_{i+1}^t) \tag{4.9}$$

where $\mathbf{E}_{w_{i+1}^t}$ denotes the embedding for the word w_{i+1}^t. An analogous computation is performed to find the probability of s_{i-1}.

Objective. Given the tuple (s_{i-1}, s_i, s_{i+1}), the objective is to optimize the sum of the log-probabilities of the next (s_{i+1}) and previous (s_{i-1}) sentences given the distributed representation ($\mathbf{h}_i^N$) of s_i:

$$\sum \log P(w_{i+1}^{t}|w_{i+1}^{<t}, h_i^N) + P(w_{i-1}^{t}|w_{i-1}^{<t}, h_i^N) \tag{4.10}$$

where the total objective is summed over all training tuples (s_{i-1}, s_i, s_{i+1}).

Sent2vec has been demonstrated to be able to infer properties of the surrounding context from just the input sentence (*e.g.*, Kiros et al. (2015)). Hence, we can assume that the distributed representations generated by *Sent2vec* also carry information regarding their context (without explicitly accessing it). Given that the surrounding context of a candidate expression is useful feature for *idiom token* classification, we can therefore train a supervised classifier only using the labelled sentences containing examples of idiomatic or literal language usage. That enables us to train such a classifier without modeling long windows of context or using methods to extract topic representations.

4.3 Experiments

In this section we describe our experiments to evaluate the predictive power of the distributed representations generated by *Sent2vec* in *idiom token* classification. We first evaluate the distributed representations using a "per-expression" design (*i.e.*, one classifier is trained for each ex-

pression) and we compare our result to those of Peng et al. (2014), that also uses a "per-expression" approach but employs multi-paragraphs of context. We then experiment with a "general" classifier trained and tested on a set of mixed expressions (*i.e.*, one single classifier is trained over a set of different expressions).

We proceed as follows: we first describe the dataset used in our experiments in Section 4.3.1; we then present the *Sent2vec* models used to generate the distributed representations in Section 4.3.2; and finally we outline the classifiers we trained in Section 4.3.3.

4.3.1 Dataset

In order to make our results comparable to those of Peng et al. (2014), we used the VNC-Tokens dataset (Cook et al., 2008). That dataset is a collection of sentences containing 53 different Verb+Noun Constructions[2] (VNCs) extracted from the British National Corpus (BNC) (Burnard, 2007). The VNC-Tokens have 2,984 sentences in total and each sentence is labelled with one of three labels: *I* (idiomatic); *L* (literal); or *Q* (unknown). Of the 53 VNCs in the dataset, 28 of them have representations of both idiomatic and literal occurrences (called by the corpus' authors "balanced portion"), and the other 25 VNCs have a skewed representation with only one class represented (called by the corpus' au-

[2]As these constructions are not analysed in isolation but within their context, we do not refer to them as Verb-Noun Idiomatic Constructions or VNICs as in Chapter 3 and we keep the original naming by the corpus' authors, Cook et al. (2008).

thors "skewed portion"). Following the same procedure taken by (Peng et al., 2014), we used the "balanced" part of the dataset and considered only those sentences labelled as *I* and *L*[3]. It is important to mention the fact that the "balanced portion" is not balanced in the traditional sense, *i.e.*, the same number of examples of both classes. In this case, the corpus' authors are using the term "balanced" to refer to the portion of the corpus that have samples of both classes independently of the proportion of samples (*i.e.*, some expressions might have a different number of positive (*I*) an negative (*L*) examples).

Peng et al. (2014) experimented with 4 different expressions from the "balanced portion" of the corpus: *BlowWhistle*; *MakeScene*; *LoseHead*; and *TakeHeart*. The authors reported Precision, Recall and F1-scores for their models trained and tested over those 4 expressions individually. In our first experiment, we compare our models with these baseline systems on a "per-expression" basis and we train a classifier for each expression individually. We follow the same distributions and methodology presented by Peng et al. (2014) and we create a training and test set for each of the 4 expressions using random sampling. In Table 4.1 we present those distributions and the split into training and test sets. The numbers in parentheses denote the number of samples labelled as *I*.

Comparing our result to those of Peng et al. (2014) has its own challenges. First, we see the choice of the 4 expressions as an arbitrary

[3] 1205 sentences in total: 749 labelled as *I*; and 456 labelled as *L*

Table 4.1
The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels (*I*) within the set. We follow the same split as described in Peng et al. (2014).

Expression	Samples	Train Size	Test Size
BlowWhistle	78 (27)	40 (20)	38 (7)
LoseHead	40 (21)	30 (15)	10 (6)
MakeScene	50 (30)	30 (15)	20 (15)
TakeHeart	81 (61)	30 (15)	51 (46)

decision to some extent as other expressions with similar ratios to those described in Table 4.1 could also have been selected for the evaluation. Moreover, the evaluation does not consider fully non-compositional expressions as the chosen expressions are all semi-compositional[4]. A better evaluation procedure would consider all 28 expressions of the balanced part of the VNC-Tokens dataset. In addition to that, we also see the choice of training and test splits as somewhat arbitrary. For example, two of the test set expressions contain samples in a way that one of the classes outnumbers the other by a great amount: the literal class of *BlowWhistle* contains roughly 4 times more samples than the idiomatic class; and the idiomatic class of *TakeHeart* contains roughly 9 times more samples than the literal class. Our main concern with these ratios is that a "per-expression" approach will achieve good performance (in terms of Precision, Recall, and F1-score) very easily over such an imbalanced distribution on the test set. Despite these concerns, in order to make a fair comparison to the baseline, we follow the expression

[4]Although we believe the task of classifying non-compositional expressions would be easier for any method aimed at *idiom token* classification as these expressions are, in general, high-fixed (see Chapter 3)

selections and training/test splits described by Peng et al. (2014).

According to studies on the characteristics of distributed representations of words in NLP (*e.g.*, Mikolov et al. (2013) and Kiros et al. (2015)), words and sentences with similar meaning tend to be represented by points close to each other in the feature space (*i.e.*, the semantic feature space). Given these properties, we follow the intuition that idiomatic phrases should also be positioned far from their literal counterparts in that space and we designed a second experiment to test whether or not *Sent2vec* would generate representations with these characteristics. In this experiment we employ the entire "balanced" part of the VNC-Tokens dataset to train and test a "general" classifier (*i.e.*, a multi-expression model).

For the experiment to be indicative of the extent to which the distributed representations position idiomatic and literal phrases in different parts of the space, we want the data to reflect as much as possible the real distribution of those expressions. Therefore, in constructing our training and test sets, we kept as much as possible the same ratio of idiomatic and literal examples, for each individual expression, across the training and test sets. With this objective in mind, we split the dataset into roughly 75% for training (917 samples) and 25% for testing (288 samples). We randomly sample the expressions ensuring that the ratio of idiomatic to literal expressions of each expression were maintained across both sets. In Table 4.2 we show the expressions used and their split into training

Table 4.2
The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels within the set.

Expression	Samples	Train Size	Test Size
BlowTop	28 (23)	21 (18)	7 (5)
BlowTrumpet	29 (19)	21 (14)	8 (5)
BlowWhistle	78 (27)	59 (20)	19 (7)
CutFigure	43 (36)	33 (28)	10 (8)
FindFoot	53 (48)	39 (36)	14 (12)
GetNod	26 (23)	19 (17)	7 (6)
GetSack	50 (43)	40 (34)	10 (9)
GetWind	28 (13)	20 (9)	8 (4)
HaveWord	91 (80)	69 (61)	22 (19)
HitRoad	32 (25)	24 (19)	8 (6)
HitRoof	18 (11)	14 (9)	4 (2)
HitWall	63 (7)	50 (6)	13 (1)
HoldFire	23 (7)	19 (5)	4 (2)
KickHeel	39 (31)	30 (23)	9 (8)
LoseHead	40 (21)	29 (15)	11 (6)
LoseThread	20 (18)	16 (15)	4 (3)
MakeFace	41 (27)	31 (21)	10 (6)
MakeHay	17 (9)	12 (6)	5 (3)
MakeHit	14 (5)	9 (3)	5 (2)
MakeMark	85 (72)	66 (56)	19 (16)
MakePile	25 (8)	18 (6)	7 (2)
MakeScene	50 (30)	37 (22)	13 (8)
PullLeg	51 (11)	40 (8)	11 (3)
PullPlug	64 (44)	49 (33)	15 (11)
PullPunch	22 (18)	18 (15)	4 (3)
PullWeight	33 (27)	24 (20)	9 (7)
SeeStar	61 (5)	49 (3)	12 (2)
TakeHeart	81 (61)	61 (45)	20 (16)

and testing data. The numbers in parentheses are the number of samples labelled as *I*.

4.3.2 *Sent2vec* Models

To encode the sentences into their distributed representations we used the code and models made available[5] by Kiros et al. (2015). With these models it is possible to encode the sentences into three different formats: *uni-skip* (which uses a regular RNN to encode the sentence into a 2400-dimensional vector); *bi-skip* (that uses a bidirectional RNN to also encode the sentence into a 2400-dimensional vector); and *comb-skip* (a concatenation of *uni-skip* and *bi-skip* which in turn has 4800 dimensions). These models were trained using the BookCorpus dataset (Zhu et al., 2015) and has been tested in several different NLP tasks such as semantic relatedness, paraphrase detection and image-sentence ranking. Given the fact that the results obtained by using either the *uni-skip* or *bi-skip* formats were far below the baselines[6], we report only the results of classifiers trained and tested using the *comb-skip* features.

4.3.3 Classifiers

4.3.3.1 "Per-expression" classifiers

The key idea behind *Sent2vec* is similar to that of distributed semantics of words: sentences containing similar meanings should be represented by points close to each other in the semantic feature space. Follow-

[5]https://github.com/ryankiros/skip-thoughts

[6]For example, the best F1-score obtained by a model using *uni-skip* was 0.16, obtained on the *LoseHead* expression. In addition, the best F1-score obtained by a model using *bi-skip* was 0.21, obtained on the *MakeScene* expression.

ing this idea, we experiment first with a similarity based classifier, the K-Nearest Neighbours (k-NN). For the k-NNs we experimented with different values for $k = \{2, 3, 5, 10\}$.

We also experimented with a classifier of the same class of algorithms as *Sent2vec* (*i.e.*, an error based classifier), the SVM (Vapnik, 1995). We trained the SVM under three different configurations:

- `Linear-SVM-PE`[7]. This model used a "linear" kernel with $C = 1.0$ on all classification setups.
- `Grid-SVM-PE`. For this model we performed a grid search for the best parameters for each expression. The parameters for each expression are reported in Table 4.3.
- `SGD-SVM-PE`. This model is a SVM with linear kernel and $C = 1.0$ but trained using stochastic gradient descent (Bottou, 2010). We set the SGD's learning rates (α) using a grid search and report it for each expression in Table 4.4. We trained each classifier for 15 epochs.

4.3.3.2 "General" classifiers

We consider the task of creating a "general" classifier that takes an example of any potential idiom and classifies it into idiomatic or literal usage to be more difficult than a "per-expression" classification

[7] `PE` stands for "per-expression"

Table 4.3
The parameters returned during the grid search for the `Grid-SVM-PE`. We set the `kernel` to be one of {`rbf`, `sigmoid`, `polynomial`} and we set $C = \{1, 10, 100, 1000\}$.

Expression	*kernel*	C
BlowWhiste	`rbf`	100
LoseHead	`rbf`	1
MakeSene	`rbf`	100
TakeHeart	`rbf`	1000

task. Hence we executed this part of the study with the SVM models only. We trained the same three types of SVM models used in the "per-expression" approach but with the following parameters:

- `Linear-SVM-GE`[8]. This model used a linear kernel with $C = 1.0$ for all the classification sets.

- `Grid-SVM-GE`. For this model we also performed a grid search and set the kernel to "polynomial" of $degree = 2$ with $C = 1000$.

- `SGD-SVM-GE`. We also experimented with a SVM with linear kernel trained using stochastic gradient descent. We set the SGD's learning rate $\alpha = 0.0001$ after performing a grid search. We trained this classifier for 15 epochs.

4.4 Results

We first present the results for the per expression comparison with Peng et al. (2014) (Section 4.4.1) and then in Section 4.4.2 we present the

[8] `smallGE` stands for "general".

Table 4.4
The learning rates (α) returned during the grid search for the `SGD-SVM-PE`. We trained each classifier for 15 epochs.

Expression	learning rate
BlowWhiste	0.001
LoseHead	0.01
MakeSene	0.0001
TakeHeart	0.0001

Table 4.5
Results in terms of Precision (P.), Recall (R.) and F1-score (F1) on *BlowWhistle* and *LoseHead*. The results of (Peng et al., 2014) are those of the multi-paragraphs method. The bold values indicates the best results for that expression in terms of F1-score.

	BlowWhistle			*LoseHead*		
	P.	R.	F1	P.	R.	F1
Peng et al. (2014)						
FDA-Topics	0.62	0.60	0.61	0.76	0.97	**0.85**
FDA-Topics+A	0.47	0.44	0.45	0.74	0.93	0.82
FDA-Text	0.65	0.43	0.52	0.72	0.73	0.72
FDA-Text+A	0.45	0.49	0.47	0.67	0.88	0.76
SVMs-Topics	0.07	0.40	0.12	0.60	0.83	0.70
SVMs-Topics+A	0.21	0.54	0.30	0.66	0.77	0.71
SVMs-Text	0.17	0.90	0.29	0.30	0.50	0.38
SVMs-Text+A	0.24	0.87	0.38	0.66	0.85	0.74
Distributed Representations						
KNN-2	0.61	0.41	0.49	0.30	0.64	0.41
KNN-3	0.84	0.32	0.46	0.58	0.65	0.61
KNN-5	0.79	0.28	0.41	0.57	0.65	0.61
KNN-10	0.83	0.30	0.44	0.28	0.68	0.40
`Linear-SVM-PE`	0.77	0.50	0.60	0.72	0.84	0.77
`Grid-SVM-PE`	0.80	0.51	**0.62**	0.83	0.89	**0.85**
`SGD-SVM-PE`	0.70	0.40	0.51	0.73	0.79	0.76

results for the "general" classifier approach.

4.4.1 Per-Expression Classification

The averaged results over 10 runs (sampling different training and test sets on each run) in terms of Precision, Recall and F1-score are presented

Table 4.6
Results in terms of Precision (P.), Recall (R.) and F1-score (F1) on *MakeScene* and *TakeHeart*. The results of (Peng et al., 2014) are those of the multi-paragraphs method. The bold values indicates the best results for that expression in terms of F1-score.

	MakeScene			*TakeHeart*		
	P.	R.	F1	P.	R.	F1
Peng et al. (2014)						
FDA-Topics	0.79	0.95	0.86	0.93	0.99	**0.96**
FDA-Topics+A	0.82	0.69	0.75	0.92	0.98	0.95
FDA-Text	0.79	0.95	0.86	0.46	0.40	0.43
FDA-Text+A	0.80	0.99	**0.88**	0.47	0.29	0.36
SVMs-Topics	0.46	0.57	0.51	0.90	1.00	0.95
SVMs-Topics+A	0.42	0.29	0.34	0.91	1.00	0.95
SVMs-Text	0.10	0.01	0.02	0.65	0.21	0.32
SVMs-Text+A	0.07	0.01	0.02	0.74	0.13	0.22
Distributed Representations						
KNN-2	0.55	0.89	0.68	0.46	0.96	0.62
KNN-3	0.88	0.88	**0.88**	0.72	0.94	0.81
KNN-5	0.87	0.83	0.85	0.73	0.94	0.82
KNN-10	0.85	0.83	0.84	0.78	0.94	0.85
`Linear-SVM-PE`	0.81	0.91	0.86	0.73	0.96	0.83
`Grid-SVM-PE`	0.80	0.91	0.85	0.72	0.96	0.82
`SGD-SVM-PE`	0.85	0.91	**0.88**	0.61	0.95	0.74

in Tables 4.5 and 4.6. When calculating these metrics, we considered the positive class to be the *I* (idiomatic) label. We applied McNemar's test (McNemar, 1947) to test for statistical significance over our results by pair-wise comparisons among all models and found all $p < 0.05$.

We can see from Tables 4.5 and 4.6 that some of our models outperform the baselines on one expression (*BlowWhistle*) and achieved the same F1-scores on the other two expressions (*LoseHead* and *MakeScene*). For theses 3 expressions, our best models generally had higher Precision than the baselines, finding more idioms on the test sets. In addition, for *MakeScene*, 2 of our models achieved the same F1-scores

(*KNN-3* and SGD-SVM-PE), although they have different Precision and Recall. The baselines performed better than ours in one expression (*Take-Heart*) across all the three metrics considered.

4.4.2 General Classification

Moving on to the "general" classifiers, we present the average results in terms of Precision, Recall and F1-score over 10 runs (sampling different training and test sets on each run) in Table 4.7. Once again, the positive class is assumed to be the *I* (idiomatic) label. We also applied McNemar's test (McNemar, 1947) to test for statistical significance over the results by pair-wise comparisons over all three models and we found all $p < 0.05$.

It should be noted that the "per-expression" evaluation was performed using a balanced set to train the classifiers while in this experiment we maintained the ratio of idiomatic to literal usages for each expression across the training and test sets. Our motivation for maintaining this ratio was to simulate the real distribution of the classes in a corpus.

We present results for the four individual expressions used in the "per-expression" evaluation as well the results for all 28 expression in the "balanced portion" of the dataset. It is important to note that, although we show the results for each expression, all three classifiers were trained and tested on the entire training and test set splits (containing all expressions) in each run. Referring to the results, we first of all note

Table 4.7
Precision (P.), Recall (R.) and F1-scores (F1) calculated on the expressions of the balanced part of the VNC-Tokens dataset. The expressions marked with * indicate the expressions also evaluated with the "per-expression" classifiers. The *Total* row lists the overall results when all expressions were considered.

	`Linear-SVM-GE`			`Grid-SVM-GE`			`SGD-SVM-GE`		
	P.	R.	F1	P.	R.	F1	P.	R.	F1
BlowTop	0.91	0.96	0.94	0.91	0.93	0.94	0.80	0.98	0.88
BlowTrumpet	0.98	0.88	0.93	0.98	0.88	0.93	0.89	0.93	0.90
BlowWhistle*	0.84	0.67	0.75	0.84	0.68	0.75	0.67	0.59	0.63
CutFigure	0.91	0.85	0.88	0.89	0.85	0.87	0.86	0.85	0.86
FindFoot	0.96	0.93	0.94	0.97	0.93	0.95	0.85	0.90	0.87
GetNod	0.98	0.91	0.95	0.98	0.91	0.95	0.91	0.91	0.91
GetSack	0.87	0.89	0.88	0.86	0.88	0.87	0.81	0.89	0.84
GetWind	0.86	0.82	0.84	0.92	0.85	0.88	0.69	0.81	0.75
HaveWord	0.99	0.89	0.94	0.99	0.89	0.94	0.95	0.91	0.93
HitRoad	0.86	0.98	0.92	0.89	0.98	0.93	0.83	0.98	0.90
HitRoof	0.88	0.88	0.88	0.92	0.88	0.90	0.80	0.83	0.82
HitWall	0.74	0.58	0.65	0.74	0.58	0.65	0.74	0.45	0.56
HoldFire	1.00	0.63	0.77	1.00	0.63	0.77	0.82	0.67	0.74
KickHeel	0.92	0.96	0.94	0.92	0.99	0.95	0.89	0.92	0.91
LoseHead*	0.78	0.66	0.72	0.75	0.64	0.69	0.75	0.67	0.71
LoseThread	1.00	0.88	0.93	1.00	0.86	0.92	0.81	0.85	0.83
MakeFace	0.70	0.83	0.76	0.69	0.76	0.72	0.62	0.81	0.70
MakeHay	0.81	0.78	0.79	0.81	0.84	0.82	0.73	0.76	0.75
MakeHit	0.10	0.54	0.70	0.10	0.54	0.70	0.85	0.55	0.67
MakeMark	0.99	0.92	0.95	0.98	0.91	0.94	0.93	0.93	0.93
MakePile	0.84	0.67	0.74	0.84	0.70	0.76	0.74	0.70	0.72
MakeScene*	0.92	0.84	0.88	0.92	0.81	0.86	0.78	0.81	0.79
PullLeg	0.79	0.71	0.75	0.82	0.72	0.77	0.75	0.70	0.72
PullPlug	0.91	0.91	0.91	0.91	0.91	0.91	0.90	0.92	0.91
PullPunch	0.85	0.87	0.86	0.87	0.87	0.87	0.70	0.85	0.77
PullWeight	1.00	0.96	0.98	1.00	0.96	0.98	0.89	0.93	0.93
SeeStar	0.17	0.13	0.15	0.17	0.13	0.15	0.17	0.17	0.17
TakeHeart*	0.94	0.79	0.86	0.94	0.80	0.86	0.86	0.80	0.83
Total	0.84	0.80	0.83	0.84	0.80	0.83	0.79	0.79	0.78

the overall performance (*Total*) of the "general" classifiers is fairly high with 2 classifiers (`Linear-SVM-GE` and `Grid-SVM-GE`) sharing the same Precision, Recall and F1-scores. While the overall results are the same for two of the classifiers, it is worth noting that deviations oc-

curred across individual expressions, although these deviations balanced out across the data set. In addition, as displayed in this table, it should be noted that all three classifiers had an extremely low performance on *SeeStar* (f1 = 0.15, 0.15 and 0.17 respectively).

If we compare the performance of the 4 expressions analysed in the "per-expression" experiment we can observe that all the "general" classifiers had a better performance over *BlowWhistle* and, on *MakeScene*, and that the `Linear-SVM-GE` also performed better than the "per-expression" classifier. We should, however, emphasize that the "general" classifier's evaluation is closer to what we would expect in a real data distribution than the evaluation presented on the "per-expression" section. This does not invalidate the evaluation of the latter but when we have access to a real data distribution it should also be taken into account when performing a ML evaluation.

4.5 Discussion

From the results we can observe that the only expression on which any of the baseline models outperformed our models was *TakeHeart* where it achieved higher Precision, Recall and F1-scores. Nevertheless, this expression had the most imbalanced test set, with roughly 9 times more idioms than literal samples. Thus, even if the baseline labelled all the test set samples as idiomatic (including the literal examples), it would still

have better performance. It is thus worth emphasizing that the choices of distributions for training and test sets by Peng et al. (2014) seems arbitrary and does not reflect the real distributions that are expected in a corpus. Also, the authors did not provide the confusion matrices for their models so we cannot analyze their model behavior across the two classes.

Regardless of that, we can say that our method is computationally cheaper than the baseline in the sense that we do not need to process any other words other than the words in the input sentence. Even though our best models share the same F1-score with the baseline on two of the tested expressions, we believe that our method is more powerful if we take into account that we do not explicitly access the context surrounding the input sentences.

Relating to the results of the "per-expression" classifiers, although the KNN-3 achieved the same F1-score as `SGD-SVM-PE` on *MakeScene*, we note that the SVMs generally outperform the KNNs and no single model perform best across all expressions. This is an interesting finding if we consider that the distributed representation is a 4,800-dimensions vector and all the SVMs are projecting that representation into a space that has far more dimensions without clear indications of the "curse of dimensionality". Previous work using *Sent2vec* has shown the capabilities of *Sent2vec* representations to capture features that are suited to various NLP tasks where semantics are involved. Those results along

with our findings suggest that the factors involved in distinguishing semantics of figurative and literal language are deeply ingrained within language generation and, thus, only a high-dimensional representation is able to capture such a distinction. Moreover, these findings also imply that the contribution of each feature (*i.e.*, each dimension of the distributed representation) is very small, given that so many dimensions are needed to unpack the components of literal and idiomatic language. Hence, there is evidence that current "manually engineered" features of previous work on *idiom token* classification are capturing a small portion of these dimensions while other dimensions are not considered. Another point for consideration is the fact that the combination of our model with the work of Peng et al. (2014) may result in a stronger model on the "per-expression" setting. Nevertheless, as previously highlighted, it was not possible for us to directly re-implement their work.

On the "general" classifier, the results are promising. It is interesting to see how the classifiers trained on a set of mixed expressions had a performance close to the "per-expression" classifiers, even though the latter were trained and tested on "artificial" training and test sets that do not reflect the real data distributions. We believe that these results indicate that the distributed representations generated by *Sent2vec* are indeed clustering together sentences within the same class (idiomatic or literal) in feature space.

On a further analysis, we concatenated the vectors for the expressions

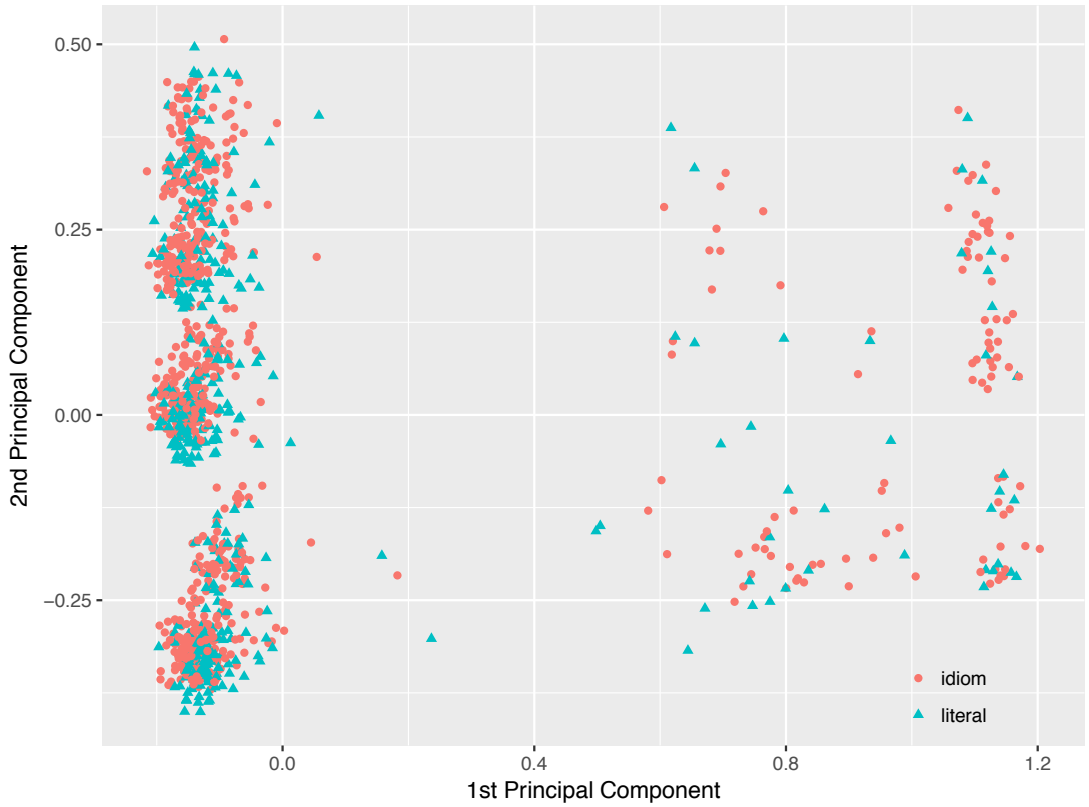


Figure 4.2
First two principal components of matrix generated by concatenating vectors generated by *Sent2vec* for each sentence used in the "general" classification experiment. On the *X*-axis we plot the first component and on the *Y*-axis we plot the second component. The circles represent sentences originally labeled as idioms (*I*) and the triangles represent sentences originally labeled as literals *L*.

into a matrix (each row is a vector generated by *Sent2vec* representing one sentence and each column is a dimension of that vector) and we applied PCA to extract the first 2 components of the resulting matrix. In Figure 4.2 we show the resulting plot of these 2 components.

Looking to Figure 4.2 we can see a separation of the idiomatic and literal representations as generated by the *Sent2vec* model used in the experiments. Although there is a large concentration over 3 clusters close to the origin of the *X*-axis, the representations of the idiomatic sentences seem to be positioned slightly above the literal sentences in

the *Y*-axis in each of the 3 clusters. With regard to the sentences that are far from the 3 clusters in relation to the *X*-axis, we can see that most of the idioms are concentrated towards the higher values relative to the *Y*-axis[9]. Although none of the clustering indicates a concentration of a particular expression, we believe this is happening due to the fact that each sentence belongs to a different subject and that the separation is purely because of the level of idiomacity of each sentence. Given this difference in the level of idiomacity, the "general" classifier is able to achieve a performance comparable to the "per-expression" classifier when considering individual expressions.

4.6 Conclusions

In this chapter we have investigated the use of distributed representations generated by Skip-Thought Vectors (*Sent2vec*) in an *idiom token* classification task. We followed the intuition that the distributed representations generated by *Sent2vec* are able to capture information regarding the context of the input sentence and therefore has sufficient information to distinguish between idiomatic and literal language use.

We tested this approach with different classifiers (k-Nearest Neighbours and Support Vector Machines) and compared our work against the state-of-the-art model that include surrounding paragraphs to the input to extract topic representations. Despite the fact that our approach

[9]On Appendix A we show the PCA plots for all 28 individual expressions used in the experiments.

takes only the sentence containing the candidate expression as its input, we have shown that by using the *Sent2vec* representations our classifiers achieve better results in 3 out of 4 expressions tested in isolation. We have also shown that our models generally present better Precision and/or Recall than the baselines.

We also investigated the capability of *Sent2vec* to cluster representations of sentences within the same class in feature space. We followed the intuition presented by previous experiments with distributed representations that sentences (and words) with similar meaning are positioned together in feature space and experimented with a "general" classifier trained on a dataset of mixed expressions. We have shown that this "general" classifier is feasible but that the traditional "per-expression" approach does achieve better results in some cases. We strongly believe that progress in understanding distributed representations will enable the development of more advanced methods suited to literal and idiomatic language and, therefore, improve the performance of the "general" classifier.

Although our approach takes only the input sentence to classify an usage of an expression as idiomatic or literal, we have demonstrated that the distributed representations generated for those sentences carries a great amount of information about the idioms contained within that sentence. Moreover, when using distributed representations, the size of the discourse history needed to generate representations for idioms

is greatly reduced while keeping the performance of the model on par with state-of-the-art approaches to *idiom token* classification. All of this underlines the potential power of the distributed representation. In the next chapter we further investigate the use of distributed representations and idioms in language modelling, a task where the representation of the discourse history changes after each input word.

Chapter 5

Attentive Language Models

Language Models (LMs) are an essential component in a range of NLP applications, such as PBSMT and Speech Recognition (Schwenk et al., 2012). An LM provides a probability for a sequence of words in a given language, reflecting the fluency and the likelihood of that word sequence occurring in that language. Traditional *n*-gram-LMs are computationally expensive given the amount of resources needed to store and process these models (Luong et al., 2015a).

The use of Recurrent Neural Networks (RNNs) have improved the state-of-the-art in LM research (Józefowicz et al., 2016) enabling the model to handle longer dependencies than traditional *n*-gram-LMs (Oualil et al., 2016). In addition to reducing the computational requirements to store and run the LM after training, the use of RNN-based LMs (RNN-LMs) is of interest for idiom research given the fact that the context of the sentence in which a candidate expression is inserted is an important feature to idiom processing (see Chapter 4). However, se-

quential data prediction is still considered a challenge in Artificial Intelligence (Mikolov et al., 2010) given that, in general, prediction accuracy degrades as the size of the sequence increases. As a result, RNN-LMs struggle in situations where there is a long-distance dependency as the relevant information from the start of the dependency has faded by the time the model has spanned the dependency. Furthermore, the context can be dominated by the more recent information so when an RNN-LM does make an error it can be propagated forward resulting in a cascade of errors through the rest of the sequence.

Recently, in sequence-to-sequence (Seq2seq) applications of RNNs, the concept of "attention" has been developed to enable RNNs to align different parts of the input and output sequences, helping the models to bridge dependencies in both sequences. Examples of these attention-based architectures include Neural Machine Translation (NMT) (Bahdanau et al., 2015; Luong et al., 2015a), image captioning (Xu et al., 2015), machine reading (Cheng et al., 2016) and sentence embeddings (Kiros et al., 2015). Inspired by this previous work in this chapter we explore the use of attention to improve the performance of RNN-LMs.

We start our exploration by introducing RNN-LMs and related research in Section 5.1. We outline our novel attention augmented RNN-LM, the *Attentive* RNN-LM model, and its advantages in Section 5.2. We describe the experimental setup and the performance of the proposed *Attentive* RNN-LM model on benchmark datasets in Section 5.3. We

then go on to further investigate the performance of the *Attentive* RNN-LM model over idiomatic language in Section 5.4. In Section 5.5 we provide an analysis and discussion about the behaviour of the *Attentive* RNN-LM. Finally, we draw our conclusions in Section 5.6.

5.1 RNN-Language Models

Typically, a "standard" RNN-LM sequentially propagates forward an internal context vector by integrating the information generated in each prediction step into the context used for the next prediction. Thus, an RNN-LM captures the probability of a sequence of words by modeling the joint probability of the words in the sequence using the chain rule:

$$p(w_1, \ldots, w_N) = \prod_{t=1}^{N} p(w_t | w_1, \ldots, w_{t-1}) \tag{5.1}$$

where N is the number of words in the sequence. The context of the word sequence is modeled by an RNN and, for each position (timestep) t in the sequence, the probability distribution over the vocabulary is calculated using a softmax on the output of the RNN's last layer (*i.e.*, the last layer's hidden state) at timestep t (Józefowicz et al., 2016):

$$p(w_1, \ldots, w_t) = softmax(\mathbf{W} h_t) \tag{5.2}$$

where $\mathbf{W}$ is a matrix of parameters and h_t is the last layer's hidden state at timestep t. In general, these models are composed of LSTM units (Hochreiter and Schmidhuber, 1997) and heavily rely on regularization to improve the RNN-LM performance (*e.g.*, Zaremba et al. (2014)). In addition, Press and Wolf (2016) also use the embedding matrix that is used to transform the input words to transform the output of the last RNN layer to feed it to the softmax layer to predict the next word in the sequence.

The attention mechanisms were first proposed in the context of "encoder-decoder" architectures for NMT systems. Bahdanau et al. (2015) proposed a model that stores all the encoder RNN's outputs and uses them together with the decoder RNN's state h_{t-1} to compute a context vector. That vector, in turn, is used to compute the current hidden state h_t. Luong et al. (2015a) present a generalization of Bahdanau et al. (2015) model that uses the decoder RNN's state, the already calculated h_t rather than h_{t-1}, along with the outputs of the encoder RNN to compute the context vector that is concatenated with h_t to make the next prediction. These models produce similar results and achieve state-of-the-art performance for some language pairs; however, they suffer from repeating words or dropping translations at the output (Mi et al., 2016). For more information on attention mechanisms applied to NMT, we refer the reader to Chapter 6.

There is also previous work on using past information to improve RNN-LMs performance to deal with long-distance dependencies. Tran et al. (2016) included memory areas at the output of every hidden layer by means of an extension to LSTM cells. These memory areas consist of two matrices of size $|V| \times d$ where $|V|$ is the size of the vocabulary and d is the number of units in the layer. Those two matrices are used to compute a distribution over previous input words and output words respectively from which the models draw the attention vector. Although the model produces good results, the model expands the number of parameters in each LSTM cell in proportion to the vocabulary size in use. In their work on Machine Reading, Cheng et al. (2016) propose to store the memory cells, instead of the outputs, of the LSTM units in every layer, at each timestep, to draw a context vector for each new input in order to attend previous content. Although the model requires fewer parameters than the model of Tran et al. (2016), when applied to language modeling its performance is below well regularized "standard" RNN-LM such as Zaremba et al. (2014) and Press and Wolf (2016).

Daniluk et al. (2017) propose an augmented version of the attention proposed by Bahdanau et al. (2015) on which their model is based on 3 vectors called *key-value-predict*. The model stores a pre-defined number of hidden states of the RNN-LM (*value* vectors) and predicts a *key*, at each timestep, to select one of the stored states, in order to retrieve information from past inputs, and generate the *predict* vector. The *pre-*

dict vector is forwarded to the softmax classifier that actually predicts the next word in the sequence. Grave et al. (2017) propose an LM augmented with a "memory cache" that stores tuples of hidden states and the embedding for the word predicted from that hidden state. The model retrieves from the memory cache the word embedding associated to the hidden state (in the memory cache) most similar to the current hidden state of the RNN-LM. Merity et al. (2017) proposed a mixture model that includes an RNN and a pointer network. This model computes one distribution for the softmax component and one distribution for the pointer network, using a sentinel gating function to combine both distributions.

Although all these models aim to use past hidden states to aid in retrieving information about past inputs, these models have a number of drawbacks. For example, the models that extend the architecture of LSTM units struggle to process large vocabularies given that the required memory expands with the size of the vocabulary. The models that retrieve a single hidden state from memory require that the prediction must be correct or otherwise the RNN-LM will not receive the correct past information. Finally, the models of Merity et al. (2017) and Grave et al. (2017) use a fixed-length memory of L previous hidden states to store and retrieve information from the past (100 states in the case of Merity et al. (2017) and 2,000 states in the case of Grave et al. (2017)). As we shall explain in Section 5.2 our *Attentive* RNN-

LM has a memory of dynamic-length that grows with the length of the input and therefore, in general, are computationally cheaper. In addition, by having a dynamic memory size, our model can fit task specific landmarks in the history, such as sentence boundaries, which makes our approach more suitable for tasks where such landmarks are important (*e.g.* sentence-by-sentence translation).

Given that our proposed model relies on the encoded information in the hidden state of the RNN-LM to represent previous input words (as we shall see in Section 5.2) and it uses a set of attention weights (instead of a *key*) to retrieve information from the past inputs, we can view our *Attentive* RNN-LM as a generalized version of these previous models. Therefore, our approach has some advantages as, for example, the reduced number of parameters given that it does not need vocabulary sized matrices in the computations of the attention mechanism. In addition, all the previous hidden states of the RNN-LM calculated for the current point in the sequence will influence the next prediction based on the calculated attention weights.

5.2 *Attentive* RNN-LMs

One consequence of the forward propagation of information in a RNN-LM is that older information tends to fade from the context as new information is integrated into the RNN context. In fact, this fading phe-

nomena occurs with almost all RNN models not augmented with an "external memory" in problems that involve long-distance dependencies. In this section we propose an extension to RNN-LMs to include an attention mechanism over the previous states within the current sequence of words the model is processing. We employ a multi-layered RNN to encode the sequence of inputs and, after each timestep t (*i.e.*, after processing each symbol of the input sequence), we store the output of the last recurrent layer ($\mathbf{h}_t$) into a memory buffer. Also, at each timestep t we compute a score for each $\mathbf{h}_i$ ($\forall\, i \in \{1, \ldots, t-1\}$) stored in memory and use the scores to weight each of those $\mathbf{h}_i$. From the weighted states we generate a context vector $\mathbf{c}_t$ that is concatenated with the current state $\mathbf{h}_t$ to predict the next word in the sequence. Figure 5.1 illustrates a step of our model when predicting the fourth word in a sequence.

To calculate the weight for each of the states and compute the context vector $\mathbf{c}_t$, we propose two different attention score functions: one, the $single(\mathbf{h}_i)$ score introduced below, is inspired by the work of Luong et al. (2015a) (*location-based* score in their paper) and calculates the attention score of each $\mathbf{h}_i$ using just the information in the state; and the other, the $combined(\mathbf{h}_i, \mathbf{h}_t)$ score described below, inspired by Bahdanau et al. (2015), calculates the attention scores for each $\mathbf{h}_i$ by combining the information from that state with the information from the current state $\mathbf{h}_t$. Each of these mechanisms define a separate *Attentive* RNN-LM and we report results for each of these LMs in our experi-

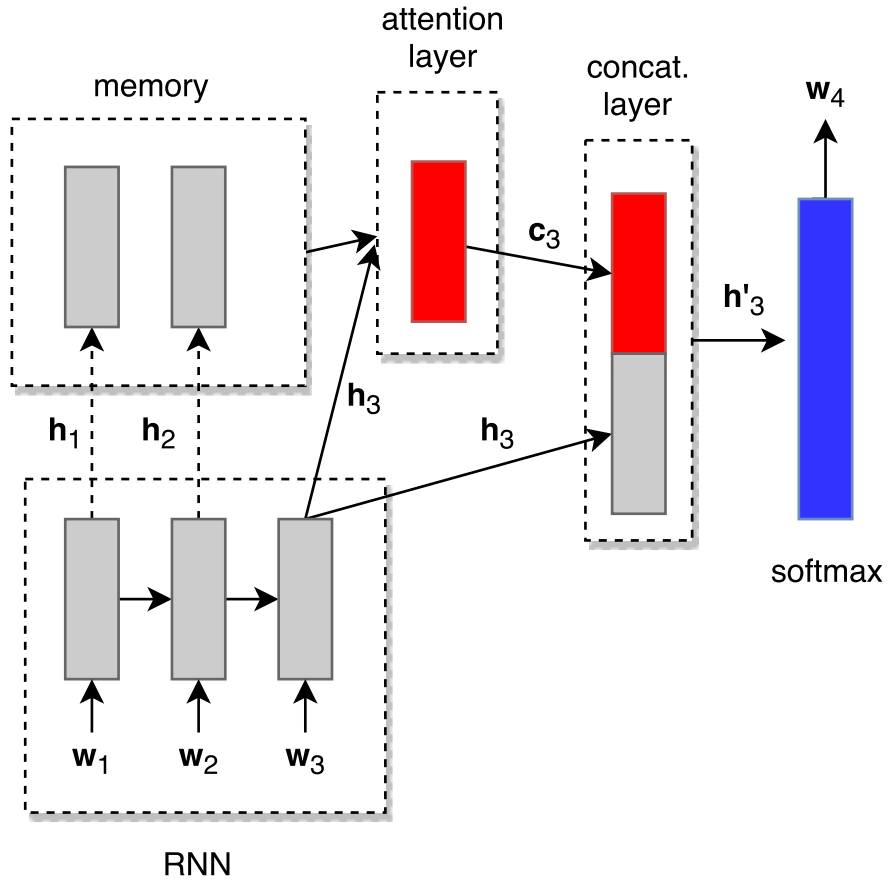


Figure 5.1
Illustration of a step of the *Attentive* RNN-LM. In this example, the model receives the third word as input ($\mathbf{w}_3$) after storing the previous states ($\mathbf{h}_1$ and $\mathbf{h}_2$) in memory. After producing $\mathbf{h}_3$, the model computes the context vector (in this case $\mathbf{c}_3$) that will be concatenated to $\mathbf{h}_3$ before the softmax layer for the prediction of the fourth word $\mathbf{w}_4$. Note that if the *single* score is in use (Eq. (5.10)), the arrow from the RNN output for $\mathbf{h}_3$ to the attention layer is dropped. Also note that $\mathbf{h}_3$ is stored in memory only at the end of this process.

ments.

More formally, each $\mathbf{h}_t$ is computed as follows, where $\mathbf{x}_t$ is the input at timestep t:

$$\mathbf{h}_t = RNN(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{5.3}$$

The context vector $\mathbf{c}_t$ is then generated using Eq. (5.4) where each scalar weight a_i is a softmax (Eq. (5.5)) and the score for each hidden state ($\mathbf{h}_i$) in the memory buffer is one of Eq. 5.6 or Eq. 5.7.

$$\mathbf{c}_t = \sum_{i=1}^{t-1} a_i \mathbf{h}_i \tag{5.4}$$

$$a_i = \frac{exp(score(\mathbf{h}_i, \mathbf{h}_t))}{\sum_{j=1}^{t-1} exp(score(\mathbf{h}_j, \mathbf{h}_t))} \tag{5.5}$$

$$score(\mathbf{h}_i, \mathbf{h}_t) = \begin{cases} single(\mathbf{h}_i) & (5.6) \\ combined(\mathbf{h}_i, \mathbf{h}_t) & (5.7) \end{cases}$$

We then merge $\mathbf{c}_t$ with the current state $\mathbf{h}_t$ using a concatenation layer[1], where $\mathbf{W}_c$ is a matrix of parameters and $\mathbf{b}_t$ is a bias vector.

$$\mathbf{h}'_t = tanh(\mathbf{W}_c[\mathbf{h}_t; \mathbf{c}_t] + \mathbf{b}_t) \tag{5.8}$$

We compute the next word probability using Eq. 5.9 where $\mathbf{W}$ is a matrix of parameters and $\mathbf{b}$ is a bias vector.

$$p(w_t|w_{<t}, x) = softmax(\mathbf{W}\mathbf{h}'_t + \mathbf{b}) \tag{5.9}$$

[1]We also have experimented with using a dot product and a feedforward layer to combine $\mathbf{h}_t$ and $\mathbf{c}_t$ and also using only $\mathbf{c}_t$, but those results were far below previous work in RNN-LM so we do not report them here.

Score single. This score is calculated for each $\mathbf{h}_i$ using just the information stored in the state itself. The score $single(\mathbf{h}_i)$ is defined as

$$single(\mathbf{h}_i) = \mathbf{v}_s \odot tanh(\mathbf{W}_s \mathbf{h}_i) \tag{5.10}$$

where the parameter matrix $\mathbf{W}_s$ and vector $\mathbf{v}_s$ are both learned by the attention mechanism and $\odot$ represents the dot product.

We expect that this type of scoring would produce a set of weights that focus on certain aspects of the sentence as, for example, the head verb. The score first performs a transformation of the state h_i using a single-layer feedforward network (the $tanh(\mathbf{W}_s \mathbf{h}_i)$ of Eq. 5.10) and then measures the similarity between the transformed state and the vector $\mathbf{v}_s$ by means of a dot product. Therefore, the vector $\mathbf{v}_s$ is expected to learn to identify the most important state within the sentence whilst the feedforward network is expected to learn how to project the states into the same space as $\mathbf{v}_s$. That most important state is likely to be the state at the timestep when the most important word (*e.g.*, the head verb) was processed by the RNN.

Given the behaviour of the score, when applying the $single(\mathbf{h}_i)$ score (which is in fact a *location-based* score), we can also think of the scalar a_i as a summary of the "absolute relevance" of the state $\mathbf{h}_i$ when in that position in the input sequence. When a new state $\mathbf{h}_t$ is added to the

buffer, its scalar summary is calculated by first using Eq.5.10 and the softmax function is applied over the set of scores calculated for each state including this new score. Although the scores for each state do not change from one timestep to the next, applying the softmax results in recalculation of the distribution of the scalar summaries for all the states $\mathbf{h}_0, \ldots, \mathbf{h}_t$. As a result the $\mathbf{a}_i$'s for each state in Eq.5.4 changes from one prediction to the next as new states are added and the weight mass is distributed across more states.

Score combined. This score is calculated for each $\mathbf{h}_i$ by combining the information from that state with the information from the current state $\mathbf{h}_t$. The score $combined(\mathbf{h}_i, \mathbf{h}_t)$ is defined as

$$combined(\mathbf{h}_i, \mathbf{h}_t) = \mathbf{v}_c \odot tanh(\mathbf{W}_s \mathbf{h}_i + \mathbf{W}_q \mathbf{h}_t) \tag{5.11}$$

where the parameter matrices $\mathbf{W}_s$ and $\mathbf{W}_q$ and vector $\mathbf{v}_s$ are learned by the attention mechanism, and $\odot$ is the same as in Eq. 5.10. Notice that because $\mathbf{W}_q \mathbf{h}_t$ does not depend on any other state and is used in the computations with all other h_i, we can compute it once and use the results in Eq. 5.11, thus reducing computation time.

When using the *combined* score, we can expect the vector $\mathbf{v}_c$ in Eq. 5.11 to have a similar behaviour to the vector $\mathbf{v}_s$ in Eq. 5.10. The main difference is that, in this score, information regarding the current state

of the network ($\mathbf{h}_t$) is added to each previous state ($\mathbf{h}_i$) before the feed-forward network projection (the $tanh(\mathbf{W}_s\mathbf{h}_i + \mathbf{W}_q\mathbf{h}_t)$ part in Eq. 5.11). Thus, the model is able to project the hidden states into different parts of the space of $\mathbf{v}_c$ at each timestep depending on the information contained in the current $\mathbf{h}_t$ of the RNN. That way, the vector $\mathbf{v}_c$ is expected to learn which hidden state is more important to the current timestep of the network (*e.g.*, the head of a prepositional-phrase when the network is processing the words in that branch) instead of focusing on the most important for the entire sequence only.

The score $\mathbf{a}_i$ defined by $combined(\mathbf{h}_i, \mathbf{h}_t)$, can also be understood as the "relative relevance" of each state $\mathbf{h}_{i<t}$ to the current state $\mathbf{h}_t$. Using this attention mechanism the score for each $\mathbf{h}_i$ is different for each timestep according to its relevance to the hidden state $\mathbf{h}_t$ (*i.e.*, the current hidden state at timestep t) of the RNN. Consequently, the scores for each $\mathbf{h}_i$ and the distribution over these scores changes from one timestep to the next. Using this scoring, the model can decide whether it should pay more attention to the current state, to a previous state or use past states to "supplement" the information for the next prediction.

In Section 5.5 we present an analysis of how the model attends to different parts of its history as it generates a sequence of predictions.

5.3 LM Experiments

To evaluate our *Attentive* RNN-LMs we conducted experiments over the PTB (Marcus et al., 1994) and wikitext2 (Merity et al., 2017) datasets. We first describe the setup of our *Attentive* RNN-LM for the PTB (Section 5.3.1) and wikitext2 (Section 5.3.2) datasets and then discuss the results (Section 5.3.3). We compare our results on PTB to Zaremba et al. (2014) and Press and Wolf (2016) the best performing LSTM-LMs on the PTB, two memory augmented networks (Grave et al. (2017) and Merity et al. (2017)) and PTB state-of-the-art ensemble models of Zaremba et al. (2014). On wikitext2 we take Merity et al. (2017), the creators of the dataset, and Grave et al. (2017), the current state-of-the-art, as our baselines.

5.3.1 PTB Setup

We evaluate our *Attentive* RNN-LM over the PTB dataset using the standard split which consists of 887K, 70K and 78K tokens on the training, validation and test sets respectively.

We follow the parameterization used by Zaremba et al. (2014) and Press and Wolf (2016) with several changes. We trained an *Attentive* RNN-LM with 2 layers of 650 LSTM units using Stochastic Gradient Descent (SGD) with an initial learning rate of 1.0, halving the learning rate at each epoch after 12 epochs, to minimize the average negative log

probability of the target words.

We train the models until we do not get any perplexity improvements over the validation set with an early stop counter of 10 epochs (*i.e.*, patience of 10 epochs). Once the model runs out of patience, we rollback its parameters and use the model that achieved the best validation perplexity to calculate the perplexity over the test set. We initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0. We also apply 50% dropout to the non-recurrent connections and clip the norm of the gradients, normalized by mini-batch size, at 5.0. In all our experiments, we follow Press and Wolf (2016) and tie the matrix $\mathbf{W}$ in Eq. (4.2) to be the embedding matrix (which also has 650 dimensions) used to represent the input words.

Contrary to Zaremba et al. (2014) and Press and Wolf (2016), we do not allow successive mini-batches to sequentially traverse the dataset. In other words, we follow the standard practice to reinitialize the hidden state of the network at the beginning of each mini-batch, by setting it to all zeros. This way, we do not allow the attention window to span across sentence boundaries[2]. We use all sentences in the training set, we truncate all sentences longer than 35 words and pad all sentences shorter than 35 words with a special symbol so that all sentences are the same

[2]We also experimented to with successive mini-batches to sequentially traverse the dataset as in Zaremba et al. (2014) but the performance of the model dropped considerably so we do not report those results here.

size. We use a vocabulary size of 10,000 words and a batch size of 32.

5.3.2 wikitext2 Setup

We also evaluate our *Attentive* RNN-LM over a larger corpus, the wikitext2 (Merity et al., 2017). We use the standard train, validation and test splits which consists of around 2M, 217K tokens and 245k tokens respectively. This dataset is composed of "Good" and "Featured" articles on Wikipedia.

There is an important difference between how we trained and tested our models on the wikitext2 dataset and how the baseline systems were trained and tested. Both Merity et al. (2017) and Grave et al. (2017) allowed the memory buffers of their systems to span sentence boundaries (and, indeed, they also did mini-batch traversal which allowed the memory buffers to traverse mini-batch boundaries) whereas we reset our systems memory at each sentence boundary. This difference is important because in the wikitext2 dataset the sentences are not shuffled and, therefore, are sequentially related to each other. As a result, systems that carry sequential information from previous sentences into the current sentence have an advantage in that they utilise contextual cues from the preceding sentence to inform the predictions at the start of the new sentence. By comparison, systems that reset their memory at the start of each sentence must reconstruct their context models from scratch and face a "cold-start" problem for the early predictions in the sentence.

The core reason why (Merity et al., 2017) and (Grave et al., 2017) did not reset their memories across sentence boundaries and we do is that these baseline systems use a fixed length memory whereas our "attention" mechanism has a variable length memory. A variable length memory has benefits in terms of both computational cost and the fact that the memory size is dynamically fitted to the context. However, just as the system designer for a fixed length memory LM must fix the memory size hyper-parameter in some fashion, so too the designer of a variable length memory LM must decide when the memory buffer is reset. For our work, we have decided to reset our memory buffer at sentence boundaries because many of the tasks for which LMs are used (*e.g.* NMT) work on a sentence by sentence basis. If required it would be possible for us to extend the memory buffer to the start of the preceding sentence (or some other landmark is the sequence history). However, this would incur extra computational cost, and as we shall see our *Attentive* RNN-LM is still competitive on the wikitext2 dataset despite the fact that the baselines systems are given access to longer context sequences.

We trained an *Attentive* RNN-LM with 2 layers of 1,000 LSTM units using SGD with an initial learning rate of 1.0, decaying the learning rate by a factor of 1.15 at each epoch after 12 epochs, to minimize the average negative log probability of the target words.

Similar to the PTB model we also train this model with an early stop counter of 10 epochs and we initialize the weight matrices of the net-

work uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0. We apply 65% dropout to the non-recurrent connections and clip the norm of the gradients, normalized by mini-batch size, at 10.0. In all our experiments, we also follow Press and Wolf (2016) and tie the matrix $\mathbf{W}$ in Eq. (4.2) to be the embedding matrix (which has 1,000 dimensions for this model) used to represent the input words. We use all sentences in the training set, but truncate all sentences longer than 35 words and pad all sentences shorter than 35 words with a special symbol so all sentences are the same length. We use a vocabulary size of 33,278 and a batch size of 32.

5.3.3 Results

In Table 5.1 we report the results of our experiments on the PTB dataset. As we can see in this table, the *Attentive* RNN-LMs outperforms all other single models on the PTB dataset. Although *Attentive* RNN-LMs have fewer parameters (10M) than the large "regularized" LSTM-LMs (66M parameters), they were capable of reducing the perplexity over both validation and test sets. This result shows that using an *Attentive* RNN-LM it is possible to achieve better perplexity scores with far fewer model parameters. Furthermore, *Attentive* RNN-LMs are able to achieve roughly the same results as the averaging of 10 RNN-LM models (with no attention) of the same size.

In addition, there is little difference between the results of the *Attent-*

Table 5.1
Perplexity results over the PTB dataset. Symbols: WT = weight tying (Press and Wolf, 2016); WD = weight decay and BD = Bayesian Dropout, both suggested by Gal and Ghahramani (2015). *combined* score indicates the context vector was calculated using Eq.5.11 and *single* score that it was calculated using Eq.5.10.

Model	Params	Valid. Set	Test Set
Single Models			
Medium Regularized LSTM (Zaremba et al., 2014)	20M	86.2	82.7
Large Regularized LSTM (Zaremba et al., 2014)	66M	82.2	78.4
Large + BD + WT (Press and Wolf, 2016)	51M	75.8	73.2
Neural cache model (size = 500) (Grave et al., 2017)	-	-	72.1
Medium Pointer Sentinel-LSTM (Merity et al., 2017)	21M	72.4	70.9
Attentive RNN-LM w/ *combined* score function	14.5M	72.6	70.7
Attentive RNN-LM w/ *single* score function	14.5M	71.7	**70.1**
Model Averaging			
2 Medium regularized LSTMs (Zaremba et al., 2014)	40M	100.4	96.1
5 Medium regularized LSTMs (Zaremba et al., 2014)	100M	76.7	73.3
10 Medium regularized LSTMs (Zaremba et al., 2014)	200M	75.2	72.0
2 Large regularized LSTMs (Zaremba et al., 2014)	122M	76.9	73.6
10 Large regularized LSTMs (Zaremba et al., 2014)	660M	72.8	69.5
38 Large regularized LSTMs (Zaremba et al., 2014)	2508M	71.9	**68.7**

ive RNN-LM with single score (Eq.5.10) and the "attentive" LM with combined score (Eq.5.11) with the single score slightly outperforming the latter. We believe this is happening because the model using the $combined(\mathbf{h}_i, \mathbf{h}_t)$ score has more parameters to optimize and, thus, more difficulties in settling to a good local optima.

In Table 5.2 we report the results on the wikitext2 dataset. Despite the fact that we reset the memory for the *Attentive* RNN-LM at each sentence boundary whereas the caches for the baseline systems span sentence boundaries, our *Attentive* RNN-LM is within 1 perplexity point of the state-of-the-art system Grave et al. (2017) (which is allowed to cache 2,000 previous hidden states), and has a lower perplexity than all

Table 5.2
Perplexity results over the wikitext2 dataset. *combined* score indicates the context vector was calculated using Eq.5.11 and *single* score that it was calculated using Eq.5.10.

Model	Params	Valid. Set	Test Set
Zoneout + Variational LSTM (Merity et al., 2017)	20M	108.7	100.9
LSTM-LM (Grave et al., 2017)	-	-	99.3
Variational LSTM (Merity et al., 2017)	20M	101.7	96.3
Neural cache model (size = 100) (Grave et al., 2017)	-	-	81.6
Pointer LSTM (window = 100) (Merity et al., 2017)	21M	84.8	80.8
Attentive RNN-LM w/ *combined* score function	50M	74.3	70.8
Attentive RNN-LM w/ *single* score function	50M	73.7	69.7
Neural cache model (size = 2000) (Grave et al., 2017)	-	-	**68.9**

of the other baselines.

5.4 Experiments with Idiomatic Language

Here our primary concern of course is to develop a LM capable of handling idioms without a degradation in its perplexity performance. In this section we describe our experiments to evaluate the *Attentive* RNN-LM over a dataset of sentences containing idioms and their literal counterparts.

5.4.1 Dataset and Baseline setup

As our idioms dataset we used the VNC-Tokens dataset (Cook et al., 2008), described in Section 4.3.1. Since, in general, an LM does not have access to a label for the sentence tagging it as idiomatic (*I*), literal (*L*) or unknown (*Q*), we selected sentences in that dataset irrespective of the label assigned to the sentence. In addition, given that an LM will

often be applied over sentences without previous knowledge of the distribution of those sentences, we do not consider the split of that dataset into "balanced portion" and "skewed portion" as we did in Section 4.3.1. Therefore, our test set of idioms contains all the 2,984 sentences of the VNC-Tokens dataset.

Idioms datasets are small to train a full LM and, in general, these models are trained on general language. Therefore, to provide insights on the performance of the *Attentive* RNN-LM, we used the same models trained in the previous section for the PTB and wikitext2 datasets and we do not perform any fine tuning or retraining.

As a baseline for the *Attentive* RNN-LM trained on the PTB, we trained a RNN-LM with two layers of LSTM units of the same size as the PTB models (650 LSTM units). This model was trained with the same hyperparameters and training procedure as the "medium size" model of Zaremba et al. (2014). The parameters were initialized uniformly in the range $[-0.05, 0.05]$ and 50% dropout was applied to the non-recurrent connections. The model had a vocabulary size of 10,000 words and was trained for 39 epochs with a learning rate of 1.0 and, after 6 epochs, the learning rate was decreased by a factor of 1.2 after each epoch. The norm of the gradients, normalized by batch size, were clipped at 5.0. This baseline model achieved a perplexity of 87.87 on the PTB validation set and a perplexity of 84.00 over the PTB test set. These results are, in comparison, only slightly worse than those reported

by Zaremba et al. (2014) and displayed in Table 5.1.

As a second baseline for the models trained over the PTB dataset, we trained another RNN-LM (RNN-LM + WT) with the exact same procedure described above but we followed Press and Wolf (2016) and tied the matrix $\mathbf{W}$ in Eq. 5.2 to be the embedding matrix (which also has 650 dimensions for this model). This RNN + WT model achieved a perplexity of 81.45 on the PTB validation set and a perplexity of 77.44 over the PTB test set. These results are, in comparison, better than the large RNN-LM models of Zaremba et al. (2014) and slightly below the large models of Press and Wolf (2016) that also use weight tying.

To compare with the *Attentive* RNN-LM models trained on the wikitext2 dataset, we trained a RNN-LM with two layers of 1,000 LSTM units. This model was trained with similar hyperparameters and training procedures as the "large size" model of Zaremba et al. (2014). The only difference is the number of LSTM units as the model of Zaremba et al. (2014) has 2 layers of 1,500 LSTM. The parameters were initialized uniformly in the range $[-0.04, 0.04]$ and 65% dropout was applied to the non-recurrent connections. The model had a vocabulary size of 33,278 words and was trained for 55 epochs with a learning rate of 1.0 and, after 14 epochs, the learning rate was decreased by a factor of 1.15 after each epoch. The norm of the gradients, normalized by batch size, were clipped at 10.0. This baseline model achieved a perplexity of 103.33 on the wikitext2 validation set and a perplexity 96.35 over the wikitext2

test set. These results are, in comparison, slightly better than those of a similar RNN-LM with LSTM units reported by Grave et al. (2017) and are similar to the results of a "Variational LSTM" model of Merity et al. (2017), both reported in Table 5.2.

We also trained a second baseline for the models trained over the wikitext2 dataset. This RNN-LM was trained with the exact procedure described for the other wikitext2 baseline but we also followed Press and Wolf (2016) and we also tied the matrix $\mathbf{W}$ in Eq. 5.2 to be the embedding matrix (which also has 1,000 dimensions for this model). This model's results are slightly better than the RNN-LM with no weight tying, but still below the memory augmented models, as reported in Table 5.2.

These results achieved by the baselines on the original test datasets indicate that both are strong baselines to compete against our *Attentive* RNN-LM over the VNC-Tokens dataset.

5.4.2 Results

In Table 5.3 and Table 5.4 we present the results in terms of perplexity over the VNC-Tokens dataset for the models trained over the PTB and the wikitext2 datasets respectively.

We can see that the *Attentive* RNN-LMs perform better than the baselines trained over both PTB and wikitext2 datasets. In the case of the models trained over the PTB dataset, both of the *Attentive* RNN-LMs

Table 5.3
Perplexity results on the VNC-Tokens dataset for models trained over the PTB dataset. *combined* score indicates the context vector was calculated using Eq.5.11 and *single* score that it was calculated using Eq.5.10. Both baselines are RNN-LMs with 2 layers of 650 LSTM units. WT = weight tying

Model	Params	PTB	VNC-Tokens
Baseline RNN-LM	16M	84.0	674.3
Baseline RNN-LM + WT	9.5M	77.4	493.8
Attentive LM w/ *combined* score function	14.5M	72.6	**338.8**
Attentive LM w/ *single* score function	14.5M	71.7	347.2

reduce the perplexity on the VNC-Tokens to a half of the perplexity obtained by the RNN-LM baseline. The difference to the the RNN-LM + WT model is smaller but still more than 150 perplexity points. In this particular scenario, the *Attentive* RNN-LM with *combined* score function outperforms the *Attentive* RNN-LM with *single* score function. This is an interesting results given that on the original PTB test set, the *single* score function had a better performance. Although the results are an improvement over the more basic RNN-LM models, there is still a large difference between the results on the idioms dataset and the original PTB test set.

Table 5.4
Perplexity results on the VNC-Tokens dataset for models trained over the wikitext2 dataset. *combined* score indicates the context vector was calculated using Eq.5.11 and *single* score that it was calculated using Eq.5.10. Both baselines are RNN-LMs with 2 layers of 1,000 LSTM units. WT = weight tying

Model	Params	wikitext2	VNC-Tokens
Baseline RNN-LM	74M	96.3	582.7
Baseline RNN-LM + WT	40.5M	91.3	561.0
Attentive LM w/ *combined* score function	50M	74.3	235.3
Attentive LM w/ *single* score function	50M	73.7	**125.0**

In the case of the models trained over the wikitext2 dataset, we can observe a similar trend. Although the difference between the worst RNN-LMs and the best *Attentive* RNN-LM in terms or perplexity points are less than a half, there is still a large gap in the results. Moreover, the gap between the worst *Attentive* RNN-LM and the best RNN-LM is increased to roughly 251 perplexity points. In addition, there is an inversion on the results of the *Attentive* RNN-LM models as the model with *single* score function is now the best model. Nevertheless, the difference between the *Attentive* RNN-LM models is smaller when trained over the wikitext2 dataset (roughly 2 perplexity points) than the difference between the models trained over the PTB dataset (around 9 perplexity points).

These results are surprising in the sense that the wikitext2 is a larger dataset of more general text than the PTB dataset which is a strictly *in domain* corpus of news text. However, the PTB dataset has a much smaller vocabulary size and, when that vocabulary is applied to the VNC-Tokens corpus, around 29% of the words are mapped to the `<UNK>` token versus 7% of words mapped to the `<UNK>` when applying the wikitext2 vocabulary. This difference in the number of *out-of-vocabulary* (OOV) words can have a significant impact on the performance of the models. However, the difference in performance of the *Attentive* RNN-LM models trained in both datasets is only 39 perplexity points from the best (trained over the wikitext2, smaller percentage of OOV words)

to the worst (trained over the PTB, larger percentage of OOV words). This suggests that the *Attentive* RNN-LM is more robust in the presence of OOV words (even when dealing with figurative language) than the baseline RNN-LM. In addition, the best performing baseline RNN-LM model over the VNC-Tokens dataset is trained on the PTB corpus and, therefore, contains a high percentage of OOV words. This percentage of OOV words makes the task somewhat easier for that model given that the model will process sentences containing a high number of OOV words and may be able to simply ignore the presence of an idiom in favour of the OOV words.

5.5 Analysis of the *Attentive* RNN-LMs

The purpose of our attention mechanism is to enable an RNN-LM to bridge long-distance dependencies in language. Therefore, we expect the attention mechanism to select previous hidden states that are relevant to the current predictions. To analyse whether the attention mechanism is functioning as we intend we analysed the distribution and evolution of attention weights in our *Attentive* RNN-LM as we calculated the perplexity for sample sentences containing long-distance dependencies (Section 5.5.1) and idioms (Section 5.5.2). In the final subsection (Section 5.5.3) we interpret the results and give some intuitions on the behaviour of the models.

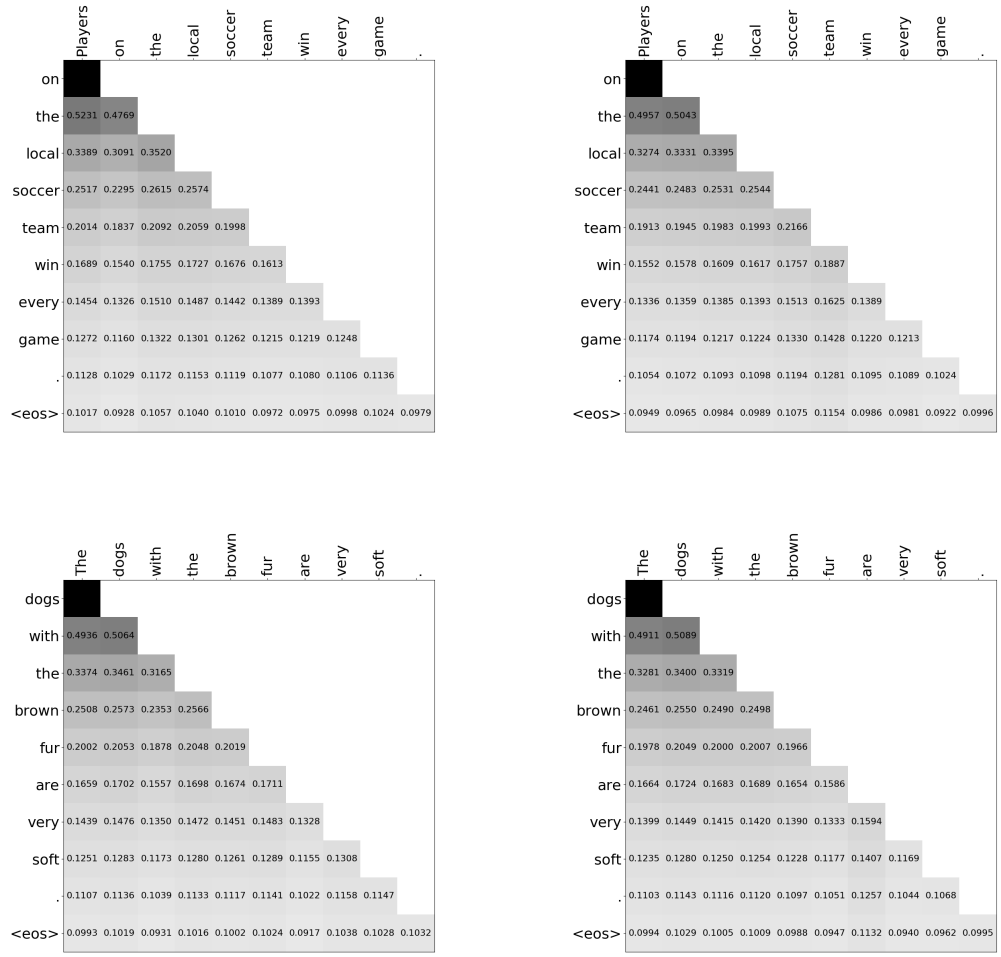


Figure 5.2
Plot of attention weights for two sentences containing nominal modifiers. On the left column are the attention weights calculated by the *combined* score. On the right column are the attention weights calculated by the *single* score. The words in the X-axis (horizontal) are the inputs at each timestep and the words in the Y-axis (vertical) are the next (or predicted) words. We suppressed weights that either equal to 1.0 (black squares) or 0.0 (white squares). Note that given the rounding to 4 decimal places, weights in some rows of the matrices may not sum to 1.0.

5.5.1 Regular Language

Figures 5.2 and 5.3 show plots of the evolution of attention weights using both *combined* score (left column of both figures) and *single* score (right column of both figures) scores when applied to regular language.

On the top row of Figure 5.2 there are plots of a sentence where the

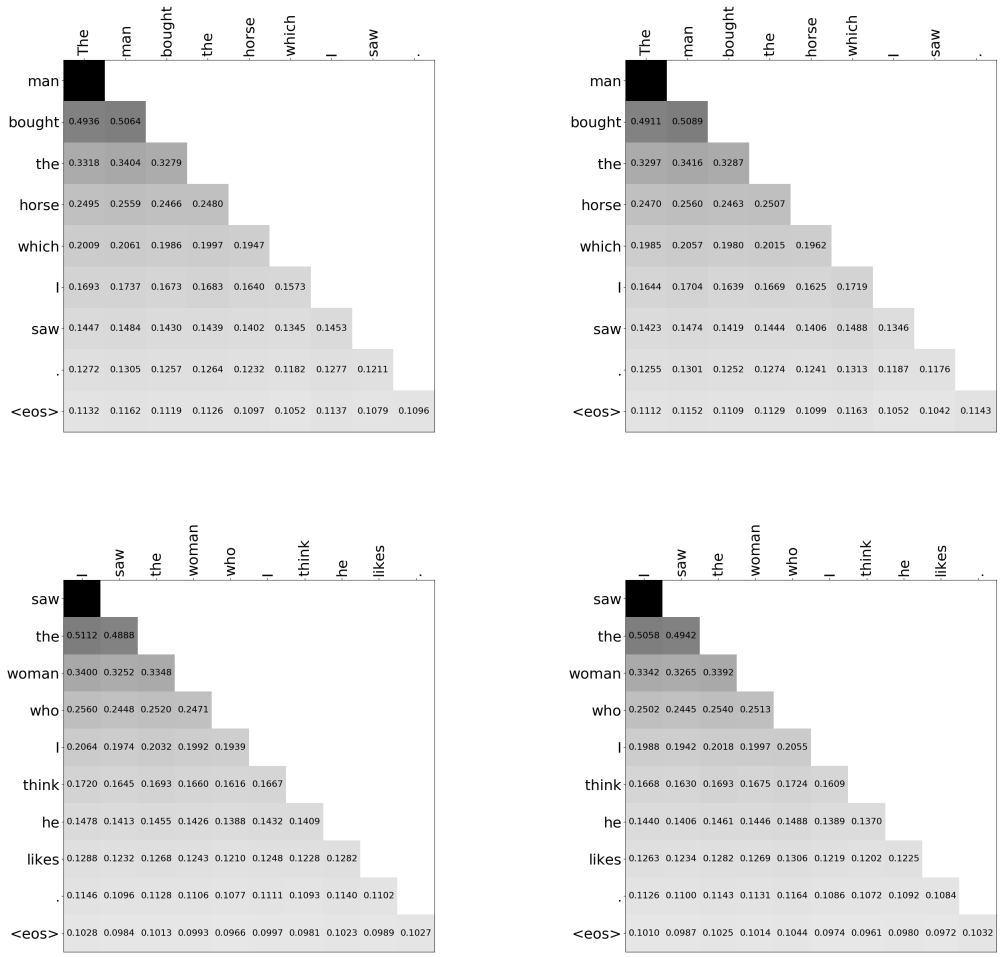


Figure 5.3
Plot of attention weights for two sentences containing relative clauses. On the left column are the attention weights calculated by the *combined* score. On the right column are the attention weights calculated by the *single* score. The words in the X-axis (horizontal) are the inputs at each timestep and the words in the Y-axis (vertical) are the next (or predicted) words. We suppressed weights that either equal to 1.0 (black squares) or 0.0 (white squares). Note that given the rounding to 4 decimal places, weights in some rows of the matrices may not sum to 1.0.

subject and the verb are separated by a nominal modifier. In this particular example, none of the models applied more attention to the subject of the sentence (*Players*) when processing the verb (*win*). The *combined* score (top left plot of the figure) had more attention on the adjective of the nominal modifier (*local*) and its determiner (*the*). The *single* score

had more attention weight on the words of the nominal modifier while keeping the smallest weight to the head verb.

On the bottom row of Figure 5.2, there are plots of a sentence where the subject is distant from the verb, separated by a nominal modifier.

Looking at the attention distribution for the two systems after the word *dog* (the subject of the sentence) has just been processed we can see that the *single* score (bottom right plot of the figure) had a slightly higher attention over that word whilst the *combined* score had its attention over the word that was processed on the immediately previous timestep (*fur*), although the attention over the verb was higher than the other words in that sentence. In this particular example, the attention generated by the *single* score was able to detect the head verb of the sentence and keep more focus over it until the <eos> symbol was processed. The attention generated by the *combined* score identified the subject and the nominal modifier keeping the attention over these parts of the sentence slightly higher than the rest of the sequence. In addition, the *combined* score was the only one of the two scores to pay attention to the preceding adverb (*very*) when processing the adjective (*soft*). In fact, the smallest attention weight given by the *single* score was on the adverb at the timestep the adjective was processed.

On the top row of Figure 5.3, there are plots of a sentence containing sentence with a relative clause. Both *combined* score (top left plot of the figure) and *single* score (top right of the image) had more attention

weight over the subject (*man*) of the first clause when processing the verb of that clause (*bought*). When processing the verb of the relative clause (*saw*), the *single* score had higher attention to the direct object (*which* that in turn refers to the *horse* in the first clause) whilst the *combined* score had a higher weight over the subject of the first clause (*man*).

On the bottom row of Figure 5.3, there are plots of another sentence containing a relative clause. Although the main verb of the sentence appears right after the subject in this sample, there are other challenges for the models. For example, there is a coreference with the subject (*I*) which is located in the relative clause. The *combined* score (bottom left plot of the image) had slightly higher attention weight over the first occurrence of the subject whilst the *single* score (bottom right plot of the image) had its highest attention weight over the direct object (*who*) of the relative clause. With regards to the second verb of clausal complement of the relative clause (in this case, the word *likes*), none of the models appear to have the attention over the important words to the clause. For example, the *combined* score had a slightly higher attention over the first occurrence of the subject in this sentence and the *single* score had its attention over the direct object (*who*) of the first verb (*think*) of the clausal complement.

5.5.2 Idioms

Figure 5.4 show the evolution of attention weights using both *combined* score (left column) and *single* score (right column) scores when calculating perplexities for three sentences containing idioms from the VNC-Tokens test set. These weights were calculated using the models trained over the wikitext dataset.

It is interesting that neither of the models seems to pay higher attention to idioms, although performing better than all the baselines over such expressions. In other words, there is no spiking in the weights when entering the idioms part of the sentence. However, there are differences in the behaviour of the models. For example, the plots in the top row of the image illustrate the weights generated for a sentence containing the idiom *blow+whistle* ("bring an illicit activity to an end by informing on the person responsible") which is separated by 2 other words. As we can see in the top left plot of the image the *combined* score was able to give slightly more weight to the states generated by the words *to* and *blow* at time the word *whistle* was predicted. This behaviour is not achieved by the *single* score (top right plot of the image) as the attention weights were almost uniformly distributed across the previous states by the time the word *whistle* was processed. In addition, the highest weight at that point was given by the model to an OOV word (represented by the symbol `<unk>`).

The same trend in behaviour is presented in the plots of the middle row of the image. The sample contains an example of the *kick+heels* idiom ("to be forced to wait for a period of time"). Once again, the words that compose the idioms are separated by a modifier (*fat*) to indicate a long wait period. The *combined* score (middle left plot of the image) gave more attention weights to the states containing the words composing that are part of the idiom, *kick* and *fat*. Once again, the *single* score (middle right plot of the image) spread the weights in an almost uniform fashion to the states prior to the processing of the word *heels*. Not even the 2 OOV words at the beginning of the sentence modified the performance of the models, which in fact, is another indication that our models are robust against OOV words.

In the bottom row are the weights for a sample containing the idiom *find+feet* ("to become familiar with and confident in a new situation"). In this particular case, once again the *combined* score model (bottom left plot of the image) had a slightly higher weight on the state when the word *found* at the timestep when the word *feet* was being processed. The *single* score surprisingly put the smallest weight to the state of the word *found* at the same timestep.

5.5.3 Discussion

To our surprise, in general, none of the attention functions worked as the expected (see Section 5.2), although the plots still show a positive trend

occurring in both models. Despite the fact that the intended behaviour for the functions only emerged on a very few examples, we can see that the functions are distributing the weights almost uniformly among the words in the samples. Thus, both models seems to take into consideration all previous states, creating a *smoothing effect*. Therefore, no single state dominates the context vector by receiving a much larger attention weight than the others. This behaviour enables the models to bring forward information from the beginning of the sentence at the time it is making a prediction.

Given this *smoothing effect*, the models do not let information fade away from the context as it progresses to subsequent steps in a sequence. At the time the model reaches the softmax classifier (*i.e.* the softmax layer), all previous information about the words that preceded the current timestep is available to the classifier. Our results over the PTB and wikitext2 test sets, that contain a range of different types of sentences, can be considered as evidence that the context is more important to the *Attentive* RNN-LM model than single words when making a prediction.

Another positive result from the *smoothing effect* is the fact that the model does not need to store information about the context of the sequence in the recurrent connections of the RNN. This behaviour of the attention functions enables the model to retrieve information from the buffer to remember past words without relying solely on the RNN's internal "memory". That way, by not relying only on the recurrence to

remember past information, the model can maximize the features extracted about an input word to the hidden states at every timestep of the model. This is an advantage over other RNN-LMs that need to both extract features and keep context regarding the sequence in its connections.

On the analysis of the sentences containing idioms, the evolution of weights in both models are a surprise given the fact that the *single* score was the best performing model in terms of perplexity. Even though the *single* score does not pay more attention to words that belong to the idioms (in the same way that the *combined* score does), the best *single* score achieved around a half of the perplexity of the best *combined* score. However, as we show in Chapter 4, the context in which an idiom is inserted is an important clue for its correct processing. Therefore, although the *single* score maintains the same behaviour as in regular language, by *smoothing* the attention weights among the hidden states in the buffer the model can retrieve more past information and make an informed decision on whether to process an idiom or regular language. This switch in behaviour is evidence that this model is robust for either figurative or literal language. Although the *combined* score seems to give slightly more attention to words that compose the idiom, its behaviour is not robust enough to process the expression correctly. In addition, the behaviour of both models is not disrupted even when there is an OOV word (`<unk>`) within the sequence, which provides some evidence for the robustness of these models in OOV situations.

Another possible way of interpreting the *smoothing effect* of the *single* score and of the *combined* score (only when applied to regular language), is that there is a reinforcement of the signal similar to a residual connections in other DNN architectures with, however, a fundamental advantage. Despite the fact that a residual connection is a shortcut to use the input word at the current timestep to reinforce its signal and, therefore, maximize the extracted features about that input, it still only considers the current input. While maximizing the extracted features, these models still rely on the recurrent connections to keep the context about the sequence up to the current timestep. The *Attentive* RNN-LM, however, uses all the previous context to achieve the same effect, bringing forward the context about the sequence without relying solely on the recurrence of the network enabling the model to maximize the extracted features about input words and without the shortcut for input words.

5.6 Conclusions

In this chapter we have investigated the performance of a novel RNN-LM model when it is used to process idioms and literal language, the *Attentive* RNN-LM. We showed that a medium sized[3] *Attentive* RNN-LM achieves better performance than larger "standard" models over the PTB. Our models also achieved performance comparable to an ensemble

[3]We adopt the terminology of Zaremba et al. (2014) and Press and Wolf (2016) when referring to the size of the RNNs.

of 10 “medium” sized LSTM RNN-LMs on the PTB. We also showed that an *Attentive* RNN-LM needs less contextual information in order to achieve similar results to state-of-the-art on the wikitext2 dataset.

In addition, we also have shown that, although our attention mechanisms does not work as initially expected by placing more attention over different parts of the sentence, the *smoothing effect* it generates enables the model to bring forward past information about the context of the sentence. Therefore, the model can maximize the amount of features extracted from an input and retrieve information from past inputs by means of the attention mechanism without the need to rely solely on the recurrence of the RNN.

Even though the model does not have access to the entire local history of an idiom (*i.e.* the sentence it is inserted in) ahead of time, the *Attentive* RNN-LM is able to reconstruct its context representation at each timestep. The model achieves this by using the attention module and its *smoothing effect*, maximizing the information extracted from each input and greatly reducing the perplexity of RNN-LMs when processing idioms. In the next chapter we investigate these distributed representations in NMT, a more complex scenario where the model has access to two different forms of context: one that represents an input sequence, potentially containing an idiom; and a second context that is not known ahead of time and the model must reconstruct it at each input, which may also contain an idiom.

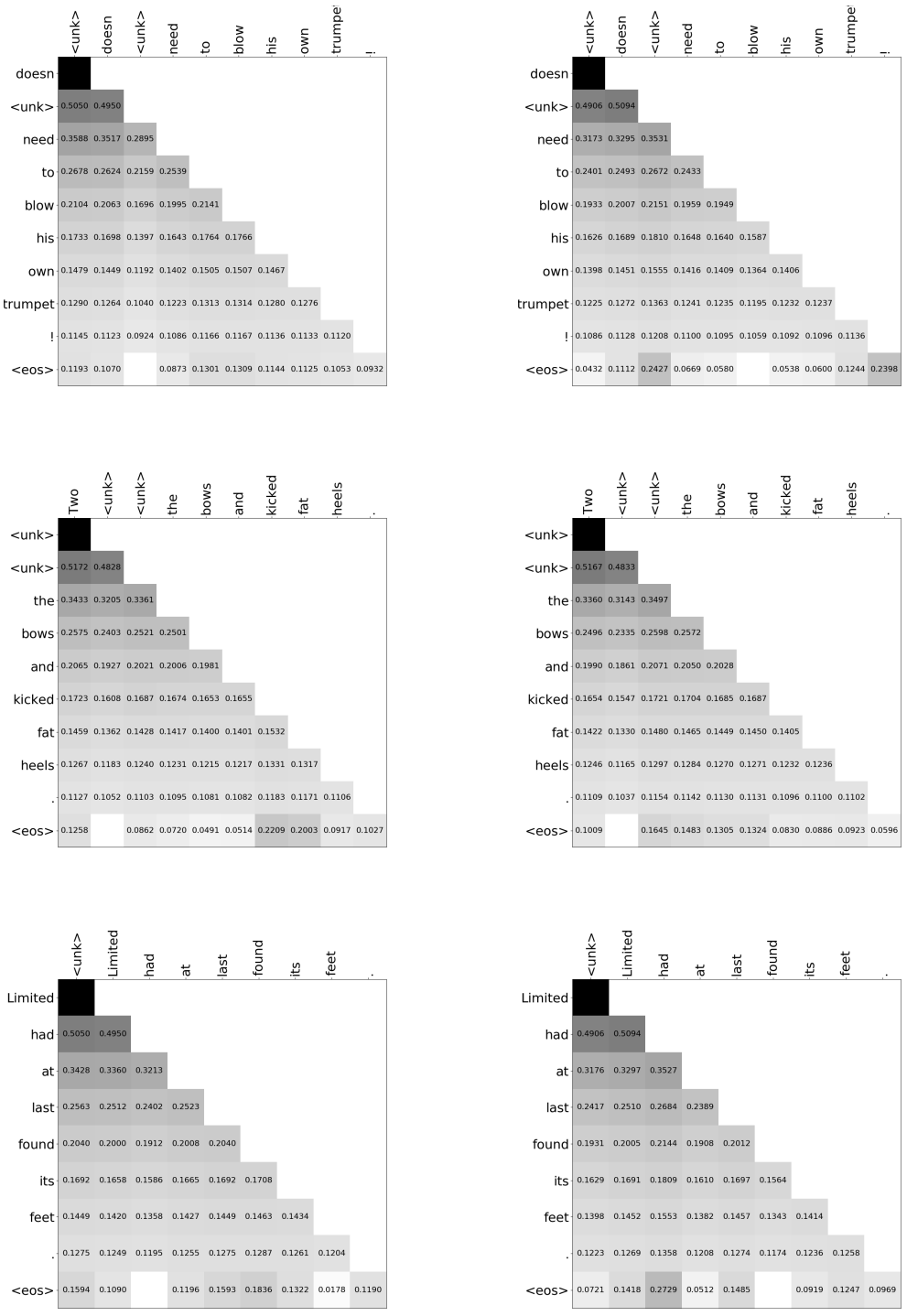


Figure 5.4
Plot of attention weights for three sentences containing idioms extracted from the VNC-Tokens dataset. On the left column are the attention weights calculated by the *combined* score. On the right column are the attention weights calculated by the *single* score. The words in the X-axis (horizontal) are the inputs at each timestep and the words in the Y-axis (vertical) are the next (or predicted) words. We suppressed weights that either equal to 1.0 (black squares) or 0.0 (white squares). Note that given the rounding to 4 decimal places, weights in some rows of the matrices may not sum to 1.0.

Chapter 6

Attentive NMT Decoder

We have shown in Chapter 2 that idioms pose problems to SMT systems as the performance of those systems (in terms of BLEU scores) is greatly reduced when translating sentences containing idioms. At the same time, many recent studies have shown that Deep Neural Networks (DNNs) can be applied to a number of NLP tasks such as speech recognition and parsing (Cho et al., 2014). DNNs are a powerful ML approach that can perform an almost arbitrary number of parallel computations for a certain number of steps enabling them to learn intricate computations (Sutskever et al., 2014).

The recent interest in Deep Learning (DL) research has resulted in ideas originating in the DNNs field being applied to Machine Translation (MT) - resulting in the development of a new area of research called Neural Machine Translation (NMT). The basic idea of NMT is to apply two different RNNs, which are a specific type of DNNs for processing sequences, in the so-called Encoder/Decoder framework (Sut-

skever et al., 2014; Cho et al., 2014). The first RNN, called the *Encoder*), is trained to compress the input sentence, written in the source language, into a distributed representation (*i.e.*, a fixed-size vector of real numbers). The second RNN, called the *Decoder*, is trained to take that distributed representation and decompress it (word-by-word) into the output sentence, written in the target language. Framed this way, the *Decoder* is essentially an RNN-LM that is conditioned on a representation of the input sentence generated by an encoder RNN.

In this chapter we present an extension to NMT that uses the *Attentive* RNN-LM (proposed in Chapter 5) as a decoder to aid the processing of sentences in which there are idioms and, also, when there are long-distance dependencies. We begin by introducing NMT in more detail and outlining previous work in the field. In Section 6.2 we describe the modifications we made to a baseline NMT architecture in order to integrate an *Attentive* RNN-LM into the architecture as the decoder within the architecture. We refer to this augmented NMT architecture as an *Attentive* NMT system. Although we are more interested in the performance of the *Attentive* NMT systems when translating idioms, we first outline our experiments to evaluate an *Attentive* NMT over benchmark datasets in Section 6.3 and use the analysis of these experiments to develop an understanding of general the behavior of an *Attentive* NMT system. We then evaluate our evaluate the performance of an *Attentive* NMT system when it is translating idioms in Section 6.4. In Section 6.5 we present

an analysis of the performance of an *Attentive* NMT system compared to a set of baselines systems on both benchmark datasets and datasets of idioms. Finally, in Section 6.6 we draw our conclusions.

6.1 Neural Machine Translation

Figure 6.1 (informally) outlines the basic NMT model. Formally, let $S = \{w_1^s, \ldots, w_{L_S}^s\}$ and $T = \{w_1^t, \ldots, w_{L_T}^t\}$ represent represent the source sentence S and the target sentence T respectively. L_S and L_T denote the lengths of S and T respectively (note that L_S might be different from L_T). Also let n_e denote the number of hidden units in the *Encoder* and n_d denote the number of hidden units in the *Decoder*.

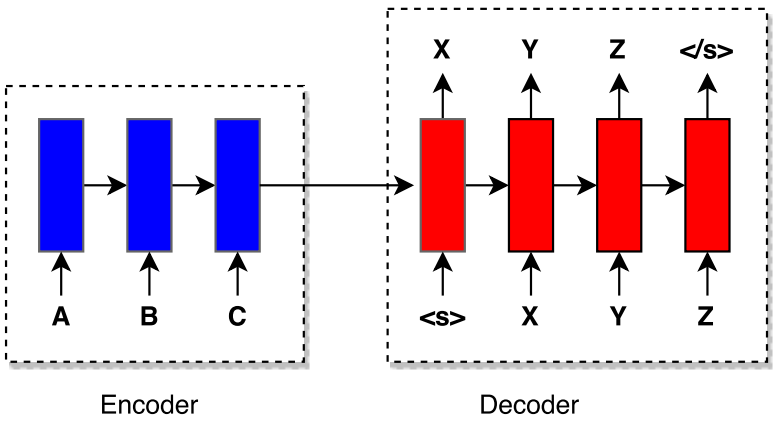


Figure 6.1
The Encoder-Decoder architecture. The rectangles inside the box on the left represent the Encoder RNN unfolded over time and the rectangles inside the box on the right represent the Decoder RNN, also unfolded over time. The output of the last step of the Encoder RNN is the distributed representation of the source sentence. The first input for the Decoder RNN is the start-of-sentence (<s>) symbol of the output sentence and the Decoder RNN's first hidden state is set to be the distributed representation of the input sentence. At each timestep t_j the Decoder RNN emits a symbol (word) that will serve as input to the Decoder RNN at timestep t_{j+1}. The Decoder RNN performs computations until the </s> symbol is emitted.

Encoder. The *Encoder* compresses information about S into a distributed representation c using a RNN. The computation of c involves iterating over the following equation:

$$\mathbf{h}_i^e = f(w_i^s, \mathbf{h}_{i-1}) \tag{6.1}$$

where f is a non-linear function; and $\mathbf{h}_i^e \in \mathbb{R}^{n_e}$ (also called the *Encoder hidden state*) is the output of f at each iteration i. The *Encoder* then outputs its last *hidden state* to be the representation $\mathbf{c}^e$:

$$\mathbf{c}^e = \mathbf{h}_{L_s}^e \tag{6.2}$$

Note that $\mathbf{c}^e \in \mathbb{R}^{n_e}$.

Decoder. The *Decoder* is often trained to predict the next word w_j^t given $\mathbf{c}^e$ and all previous $w_{<j}^t$ (*i.e.*, all words of previous timesteps to w_j^t). Therefore, the *Decoder* is understood to define a probability over the translation T using the (ordered) conditionals:

$$P(T) = \prod_{j=1}^{L_T} P(w_j^t | \{w_1^t, \ldots, w_{j-1}^t\}, \mathbf{c}^e) \tag{6.3}$$

where the probability $P(w_j^t|\{w_1^t,\ldots,w_{j-1}^t\},\mathbf{c}^e)$ is defined as:

$$P(w_j^t|\{w_1^t,\ldots,w_{j-1}^t\},\mathbf{c}^e) = g(w_{j-1}^t,\mathbf{h}_j^d) \tag{6.4}$$

where g is a non-linear function; and $\mathbf{h}_j^d \in \mathbb{R}^{n_d}$ (also called the *Decoder hidden state*) is the output of g at each iteration j.

The *Sequence-to-Sequence Learning* (*Seq2seq*) model proposed by (Sutskever et al., 2014) closely follows the *Encoder-Decoder* approach using Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units. A single *Seq2seq* model has achieved results close to the state-of-the-art SMT systems (Sutskever et al., 2014). Despite being relatively simple in comparison with other NMT systems, the ensemble of *Seq2seq* models using LSTM units still achieves the state-of-the-art results for French to English translation (Sutskever et al., 2014). Nevertheless, a negative point of these models is the fact they are difficult to train given the required number of hidden layers for each neural network and the size of the models. In fact, these models can easily have more then 26 billion parameters.

An alternative NMT model incorporating an attention mechanism was proposed by Bahdanau et al. (2015). Their model, called *NMT with Soft Attention* (or *NMT with Global Attention*), also builds upon

the *Encoder-Decoder* approach and adds a small Neural Network that learns which part of the encoded distributed representation of the source sentence to pay attention to at the different stages of the decoding process. This model is more complex then *Seq2seq* but requires a smaller number of hidden layers and parameters. This reduction in hidden layers and parameters is due to the use of Gated Recurrent Units (GRU) (Cho et al., 2014) in the hidden layers of this model. This approach was the first pure NMT system to win a Machine Translation Shared Task in the Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2015). Moreover, this system won the competition for three different language pairs (German/Czech to English, Chinese to English and Turkish to English). Figure 6.2 shows an informal representation of the *NMT with Global Attention*.

More recently, (Luong et al., 2015a) proposed the *NMT with Local Attention* model, building upon the ideas of both *Seq2seq* and *NMT with Global Attention*. In this work, (Luong et al., 2015a) also use two stacks of LSTM units (similar to *Seq2seq*) and includes two feedforward networks. The first network is trained to predict a fixed-size window over the distributed representation of the source sentence. The second network is trained to learn which part of the predicted window to pay attention to at different stages of the decoding step, similar in spirit to *NMT with Global Attention*. The *NMT with Local Attention* model is currently the state-of-the-art for translating from English into German

(Luong et al., 2015a). In addition, Luong et al. (2015a) also experimented with the use of stacks of LSTM units together with a *Global attention* mechanism (more general than the original), with slightly worse results than the *Local Attention* model. Figure 6.3 shows an informal representation of the *NMT with Local Attention*.

In addition to the architecture of the model itself, there has also been

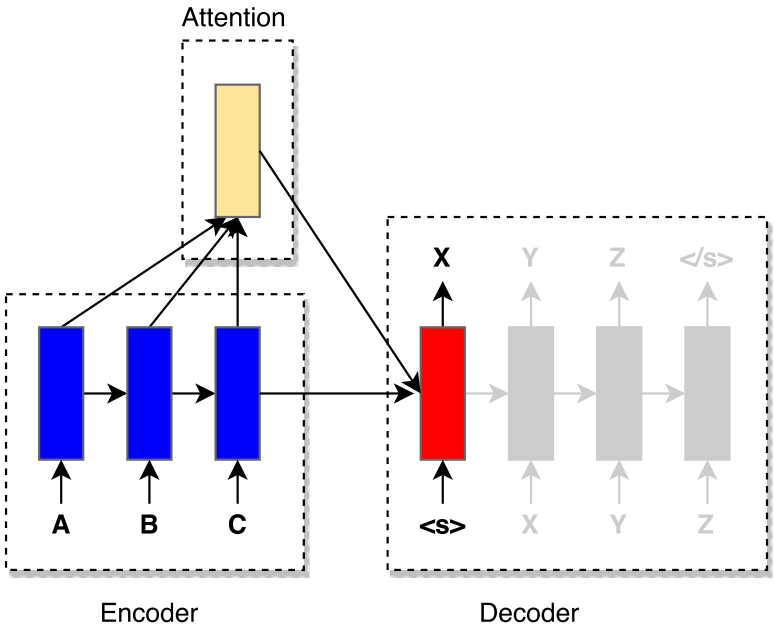


Figure 6.2
The Encoder-Decoder architecture with Global Attention. The rectangles inside the box on the left represent the Encoder RNN unfolded over time and the rectangles inside the box on the right represent the Decoder RNN, also unfolded over time. Here we outline the first timestep of the Decoder RNN whilst the reminder timestep (shaded) follow the same computation path. At each timestep t_j, the Decoder RNN receive as inputs its hidden state at t_{j-1} and the word emitted also at t_{j-1} (in the case of the first timestep outlined in the image, the Decoder RNN receives the distributed representation of the input sentence (Encoder RNN's last hidden state) and the <s>symbol). At each timestep t_j, the attention layer (rectangle inside the "("Attention) box) generate a context vector based on all hidden states of the Encoder RNN that is also fed to the Decoder RNN. The Decoder RNN emits a new symbol (word) that serves as input to the Decoder RNN at timestep t_{j+1} together with its hidden state at timestep t_j and the newly generated context vector. The Decoder RNN performs these computations until the </s> symbol is emitted.

work on addressing the translation of out-of-vocabulary (OOV) words by NMT systems. In general, these systems are trained with a smaller vocabulary than is actually found in the corpus given that the complexity (and therefore the number of parameters) increases with the size of the vocabulary (Jean et al., 2015a). With that in mind, Luong et al.

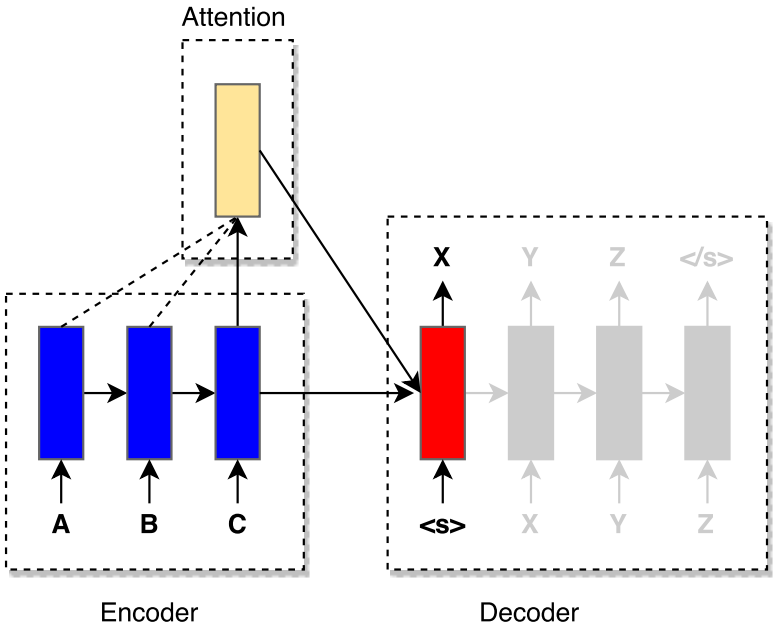


Figure 6.3
The Encoder-Decoder architecture with Local Attention. The rectangles inside the box on the left represent the Encoder RNN unfolded over time and the rectangles inside the box on the right represent the Decoder RNN, also unfolded over time. Here we outline the first timestep of the Decoder RNN whilst the reminder timestep (shaded) follow the same computation path. At each timestep t_j, the Decoder RNN receive as inputs its hidden state at t_{j-1} and the word emitted also at t_{j-1} (in the case of the first timestep outlined in the image, the Decoder RNN receives the distributed representation of the input sentence (Encoder RNN's last hidden state) and the <s>symbol). At each timestep t_j, the attention layer (rectangle inside the "("Attention) box) generate a context vector based on part of the hidden states of the Encoder RNN that is also fed to the Decoder RNN. In this particular representation, the Attention Layer choose to use only the last hidden state of the Encoder RNN to generate the attention context. The Decoder RNN emits a new symbol (word) that serves as input to the Decoder RNN at timestep t_{j+1} together with its hidden state at timestep t_j and the newly generated context vector. The Decoder RNN performs these computations until the </s> symbol is emitted.

(2015b) propose to extract a bilingual dictionary based on the alignment of words in the aligned corpus used to train the NMT model. Once the translations are obtained, their model replace all the (OOV) words in the source sentence by its correspondent in the extracted dictionary. Jean et al. (2015a) adopts an approach using "importance sampling" to deal with OOV words. In their work, the authors keep all words in the vocabulary and, during the backpropagation step used to train the model, they sample a certain numbers of words to compute the normalization constant of the gradient's expectation. Therefore, at each parameter update, only the parameters corresponding to the correct word and to the sampled words are updated. Before decoding, the authors use a subset of the K most frequent words and extract at most K' target words based on word alignments found in the corpus (similarly to Luong et al. (2015b)). During decoding the system effectively uses a vocabulary of K source and K' target words. These approaches have in common the fact that they assume a 1-to-1 mapping between OOV words in the input and words in the output. However, in some cases the word mapping might be 1-to-many and, thus, the translation performance will degrade in those cases.

On a different approach to OOV words, Sennrich et al. (2016b) propose to use Byte-Pair Encoding (BPE) (Gage, 1994), a data compression technique. When used in data compression, BPE iteratively replaces the most frequent pair of bytes in a sequence by a single, (unused) byte.

The technique is adapted by Sennrich et al. (2016b) to merge sequence of characters and, thus, frequent *n*-grams are merged into a single symbol. After applying BPE to the corpus, the final symbol vocabulary is then used as the vocabulary to train the model. After decoding, the BPE operations are reversed in order to obtain the original words. The main advantage of this technique is the fact that there is no assumption about mappings between OOV words at the input and words at the output. In addition, the use of BPE has been shown to improve translation of morphologically rich languages (Sennrich et al., 2016b).

6.2 *Attentive* NMT

As pointed out by Sutskever et al. (2014), the decoder RNN of a NMT system is essentially a RNN-LM conditioned on an input sequence encoded by another RNN. To adapt our *Attentive* RNN-LM to work as the decoder of a NMT system, we use the sequence-to-sequence model of Luong et al. (2015a).

Luong et al. (2015a) adds a memory buffer where all the hidden states $h^e_1, \ldots, h^e_{L_S}$ of the *Encoder* RNN are stored. The authors then modify Eq. 6.1 so that a new representation $\mathbf{c}^e$ for the input sequence is generated at each timestep t using an attention based model. More formally, generating c^e_t involves iterating over the following equations:

$$\mathbf{c}_t^e = \sum_{i=1}^{L_S} a_i \mathbf{h}_i^e \tag{6.5}$$

$$a_i = \frac{exp(score(\mathbf{h}_i^e, \mathbf{h}_t^d))}{\sum_{j=1}^{L_S} exp(score(\mathbf{h}_j, \mathbf{h}_t^d))} \tag{6.6}$$

$$score(\mathbf{h}_i^e, \mathbf{h}_t^d) = \begin{cases} \mathbf{h}_i^e \odot \mathbf{h}_t^d & \text{dot} \quad (6.7) \\ \mathbf{h}_i^e \odot \mathbf{W}_a \mathbf{h}_t^d & \text{general} \quad (6.8) \\ \mathbf{W}_a[\mathbf{h}_i^e; \mathbf{h}_t^d] & \text{concat} \quad (6.9) \end{cases}$$

where $\odot$ is a dot product and $\mathbf{W}_a$ is a matrix of parameters.

The vector c_t^e is then merged with the current state $\mathbf{h}_t^d$ by means of a concatenation layer

$$\mathbf{h}_t'' = tanh(\mathbf{W}_c[\mathbf{h}_t^d; c_t^e] + \mathbf{b}_t) \tag{6.10}$$

where $\mathbf{h}_t''$, in the original model, is then passed to the softmax layer to make the prediction for the next word.

To adapt our *Attentive* RNN-LM we add another concatenation layer so that we may merge $\mathbf{h}_t''$ with $\mathbf{h}_t'$ as generated following Equation 5.8:

$$\mathbf{h}_t^\dagger = tanh(\mathbf{W}[\mathbf{h}_t''; \mathbf{h}_t'] + \mathbf{b}) \tag{6.11}$$

where $\mathbf{W}$ is a matrix of parameters and $\mathbf{b}$ is the bias vector of the concatenation layer. $\mathbf{h}_t^{\dagger}$ is then passed to the softmax layer to make the next prediction.

6.3 NMT Experiments

Although we are interested on the effectiveness of the *Attentive* NMT when translating idioms, we first evaluate the effectiveness of the representations generated by this model on datasets of literal language so as to get insights about its performance. To the best of our knowledge, this is the first attempt to place attention on both the encoder states and the previous states of the decoder in an NMT system. In the next sections, we describe the setup of our *Attentive* NMT for English/Brazilian-Portuguese and English/German (Section 6.3.1 and Section 6.3.2 respectively) and then discuss the results (Section 6.3.3).

6.3.1 NMT Setup for English/Brazilian-Portuguese

We conducted experiments using the language pair English/Brazilian-Portuguese (EN/PT-BR) in both directions using the Fapesp-V2 corpus, which is split into training (160K sentences, roughly 4 times larger than the PTB dataset), development (1375 sentences), and 2 test sets (test-a, 1314 sentences and test-b, 1447 sentences).

We use *Attentive* NMT models of similar sizes in comparison to those

proposed by Luong and Manning (2015). More specifically, the encoder is a RNN composed of two layers of 1,000 LSTM units and the decoder is one of our *Attentive* RNN-LMs (also composed of two layers of 1,000 LSTM units). We also applied attention over the encoder outputs using the "dot" and "general" content functions (Eq. 6.7 and Eq. 6.8 respectively) as described in Luong et al. (2015a). In total, we have trained four models for each direction (eight in total).

We optimize the model using ADAM (Kingma and Ba, 2014) with a learning rate of 0.0001 and mini-batches of size 32 to minimize the average negative log probability of the target words. We train the models until we do not get any improvements on negative log probability over the development set with an early stop counter of 10 epochs. Once the model runs out of patience, we rollback its parameters and use the model that achieved the best performance on the validation set to obtain the translations[1]. We initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value of 0.0. We also apply 50% dropout to the non-recurrent connections and clip the norm of the gradients, normalized by mini-batch size, at 5.0. In all our models, as in the LM experiments, we also tie the matrix $\mathbf{W}$ in Eq. 4.2 to be the embedding matrix (which also has 1,000 dimensions) used to represent the input words.

[1]In this early stop setting, each model takes around 2 and a half days (depending on the number of epochs) to train on an Tesla K40 GPU. In general, the average number of epochs is around 22, including the 10 epochs of patience.

As in the LM experiments (Section 5.3) we do not allow successive mini-batches to sequentially traverse the dataset. We remove all sentences in the training set larger than 50 words and we pad all sentences shorter than 50 words with a special symbol so they will all have the same size. We use a vocabulary of the 50,000 most frequent words in the training set including three symbols to account for the padding of shorter sentences, the end of sequence and OOV words respectively.

NMT Baseline. We used the model of Bahdanau et al. (2015)[2] as our NMT baseline. We followed the parametrization (word embeddings of 620 dimensions and 1,000 GRU units (Cho et al., 2014) in both encoder and decoder) and we followed the training procedure described by the authors to train the model: the recurrent weight matrices were initialized as random orthogonal matrices; the bias vectors were initialized at 0; the parameters of the attention layer (the encoder attention layer) were sampled from a Gaussian distribution with 0 mean and variance 0.001^2; and the remaining weight matrices were sampled from a Gaussian distribution with 0 mean and variance 0.01^2; the model was trained using Adadelta (Zeiler, 2012) with a learning rate of 0.0001 (and a "patience" of 10 epochs) and the norm of the gradients (normalized by mini-batch size) were clipped at 1.0.

SMT Baseline. For comparison we trained a PBSMT system using the Moses toolkit (Koehn et al., 2007) with its baseline settings. We used

[2]Code available at `https://github.com/nyu-dl/dl4mt-tutorial/`

a 5-gram LM trained with the KenLM toolkit (Heafield, 2011) with *modified-kneser-ney* smoothing. We used the development set to tune this model.

6.3.2 NMT Setup for English/German

We also conducted experiments using the language pair English/German (DE/EN) in both directions. We used the modified data of the WMT' 2016 shared task pre-processed as described in Section 6.4 to train both our models and the baseline models[3].

For this language pair we use *Attentive* NMT models of similar sizes in comparison to those proposed by Luong et al. (2015a). More specifically, the encoder is an RNN composed of four layers of 1,000 LSTM units and the decoder is one of our *Attentive* RNN-LMs (also composed of four layers of 1,000 LSTM units). We also applied attention over the encoder outputs using the "dot" and "general" content functions (Eq. 6.7 and Eq. 6.8 respectively) as described in Luong et al. (2015a). In total, we have trained four models for each direction (eight in total).

We optimize the *Attentive* NMT models using the same procedure and parameters as for the *Attentive* NMT models used in the experiments with the English/Brazilian-Portuguese language pair. The only modifications, given the larger size of the training set used in this experiment in comparison with the English/Brazilian-Portuguese language pair, were:

[3] In other words, we removed from the training corpora part of the sentences containing idioms from the dataset of Fritzinger et al. (2010).

(a) we used mini-batches of size 128; and (b) we used an early stop counter of 5 epochs[4].

NMT Baseline. We used the model of Sennrich et al. (2016a)[5] as our NMT baseline. In fact, this model is equal to the model of Bahdanau et al. (2015) that we use as a baseline for English/Brazilian-Portuguese apart from the fact that it uses BPE to reduce the size of the vocabulary. We followed the same parametrization (word embeddings of 500 dimensions and 1,024 GRU units in both encoder and decoder) and we followed the training procedure described for the English/Brazilian-Portuguese NMT baseline (see Section 6.3.1).

SMT Baseline. We trained a PBSMT system using the Moses toolkit (Koehn et al., 2007) with its baseline settings. Again, we used a 5-gram LM trained with the KenLM (Heafield, 2011) toolkit with *modified-kneser-ney* smoothing. In this csase, we used the *newstest2013* (3K sentences) set to tune this model.

6.3.3 NMT Results

Table 6.1 lists the results in terms of BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores for the machine translation experiments using the English/Brazilian-Portuguese language pair (in both directions).

[4]In this early stop setting, each model takes around 2 and a half weeks (depending on the number of epochs) to train on an Tesla K40 GPU. In general, the average number of epochs is around 14, including the 5 epochs of patience.

[5]Code available at `http://data.statmt.org/rsennrich/wmt16_systems/`

Table 6.1
Results in terms of BLEU and TER score on the Fapesp-v2 test sets (test-a and test-b). "Dot" and "General" refer to the content function applied to compute the attention over the encoder states. Larger BLEU and smaller TER scores indicates better performance.

Model	test-a		test-b	
	BLEU	TER	BLEU	TER
English/Brazilian-Portuguese (EN/PT-BR)				
SMT	43.5	**34.5**	34.7	**43.4**
NMT	41.7	40.0	32.2	49.7
Dot & *Attentive* NMT w/ *single* score function	44.9	36.8	35.3	46.1
Dot & *Attentive* NMT w/ *combined* score function	**46.9**	36.7	**36.5**	45.2
General & *Attentive* NMT w/ *single* score function	34.0	46.8	24.5	55.8
General & *Attentive* NMT w/ *combined* score function	35.6	45.4	25.7	55.2
Brazilian-Portuguese/English (PT-BR/EN)				
SMT	**44.5**	**34.4**	35.1	**44.1**
NMT	26.6	55.4	22.4	61.2
Dot & *Attentive* NMT w/ *single* score function	43.6	37.7	35.6	45.9
Dot & *Attentive* NMT w/ *combined* score function	**44.5**	37.6	**36.1**	46.0
General & *Attentive* NMT w/ *single* score function	31.9	47.7	24.3	56.6
General & *Attentive* NMT w/ *combined* score function	32.0	49.8	24.5	57.6

The results show that for the English/Brazilian-Portuguese direction our Dot & *Attentive* NMT w/ *combined* score function outperforms the benchmark NMT and baseline SMT systems when BLEU scores are considered on both test sets (test-a and test-b); however, the SMT system has the best performance when TER is considered.

On the direction Brazilian-Portuguese/English, a similar trend is observed. When BLEU scores are considered our Dot & *Attentive* NMT w/ *combined* score outperforms the NMT baseline on both test sets. On test-a, our best model and the SMT achieve the same performance on BLEU scores with our system outperforming the SMT on test-b. However, when TER scores are considered the SMT performs best.

Table 6.2
Results in terms of BLEU and TER score on the WMT' 2014 and WMT' 2015 test sets. "Dot" and "General" refer to the content function applied to compute the attention over the encoder states. Larger BLEU and smaller TER scores indicates better performance.

Model	WMT' 2014		WMT' 2015	
	BLEU	TER	BLEU	TER
German/English (DE/EN)				
SMT	**21.6**	**58.6**	**22.9**	**56.9**
NMT	17.9	63.3	17.7	63.9
Dot & *Attentive* NMT w/ *single* score function	14.2	70.1	14.8	68.1
General & *Attentive* NMT w/ *single* score function	13.9	70.8	14.6	68.4
English/German (EN/DE)				
SMT	14.8	69.4	17.3	**66.1**
NMT	13.6	72.3	13.8	72.4
Dot & *Attentive* NMT w/ *single* score function	15.3	**67.1**	17.2	64.8
General & *Attentive* NMT w/ *single* score function	**16.1**	68.7	**17.8**	66.2

Table 6.2 show the results of our experiments in the English/German language pair (in both directions) in terms of BLEU and TER scores. The *Attentive* NMT models with the *combined* score (either using *dot* or *general* encoder content functions) did not converged during the training phase for this language pair. In all configurations tried for these models, the negative loss likelihood started to increase beyond the initial levels around the middle of the second or third epochs without decreasing again as is often the case with RNNs. In fact, we tried to vary the dropout rate and the threshold when clipping the gradients norm for these models without any success. The translations obtained by such models were similar to random noise and we halted these experiments. Thus, we do not report the results for the *Attentive* NMT models with the *combined* score models in this table.

Although the baseline NMT is the model that won the WMT' 2016 shared task on the English/German language pair, on both directions, the results obtained in our experiments were below the SMT baseline for both datasets (WMT' 2014 and WMT' 2015) and both metrics (BLEU and TER). However, the entries submitted by Sennrich et al. (2016a) were the results obtained by an ensemble of these baseline models, including reranking and other post-processing steps on the output. As we are interested in the comparisons of single models, we do not apply the same steps as they did.

Despite the good performance over the English/Brazilian-Portuguese language pair, neither of our *Attentive* NMT with *single* score models had the same level of performance in the English/German direction. The results obtained by the *Attentive* NMT with *single* score and both encoder content functions were below the baselines, *i.e.* NMT and SMT.

In the English/German direction, General & *Attentive* NMT w/ *single* score function achieve the highest BLEU score on both WMT' 2014 and WMT' 2015 test sets. However, as happened in the English/Brazilian--Portuguese experiments, the best model in terms of BLEU is not the best model in terms of TER. Using this metric to score the systems, the *Attentive* NMT with *single* score and *dot* encoder content function had a better performance on WMT' 2014 while the baseline SMT performed better on the WMT' 2015.

We provide a more in depth analysis and discussion of our intuitions

about the results for both language pairs in Section 6.5.

6.4 Experiments with Idiomatic Language

As we have shown in Chapter 2, the performance of an SMT system degrades when there are idioms present in the input sentence to the system. However, we also have shown in Chapter 5 that an *Attentive* RNN-LM is better able to handle idioms (in terms of measured perplexity) compared with baseline RNN-LMs. In addition, in Section 6.3 we have demonstrated that an *Attentive* NMT can achieve higher performance scores compared to standard NMT systems when translating literal language for some language pairs. In addition, we show that the *Attentive* NMT can improve the translation of literal language for some language pairs in Section 6.3. Building on these results, in this section we focus on estimating the the ability of an *Attentive* NMT system on idiom translation.

Datasets and setup for English/Brazilian-Portuguese. When training our English/Brazilian-Portuguese NMT systems and both baselines (NMT and SMT), we used the Fapesp-v2 corpus (Aziz and Specia, 2011), which is a high quality corpus and the standard corpus to measure performance for that language pair. Given the fact that the Fapesp-v2 is based on scientific text (see Section 2.3.1), we must create a new dataset of *in-domain* idioms to evaluate.

To build this dataset of *in-domain* idioms, we used the "general" classifier for *idiom token* identification that we presented in Chapter 4. We applied the Skip-Thought Vectors to extract features for all sentences from the English side of the test-a and test-b corpora of the Fapesp-v2 and we used the `Linear-SVM-GE` classifier to identify sentences containing idioms. Given that the Skip-Thought Vectors and `Linear-SVM-GE` were trained in an English corpora of general language and the Fapesp-v2 contains some Brazilian-Portuguese words and other Brazilian-Portuguese proper names, the model classified every sentence containing those words as idiomatic. We then removed from that set all sentences containing Brazilian-Portuguese words and, after this filtering step, the dataset contained 31 sentences. We further checked this dataset and we found that only 23 sentences contained idioms (and therefore 8 sentences were literal, *i.e.*, did not contained any idioms), including VNICs and phrasal verbs. The inclusion of phrasal verbs in the dataset is a strong indication of the suitability of our "general" approach to *idiom token* identification.

Given that the number of idioms identified in Fapesp-v2 is small, we decided to include the dataset of high and low fixed idioms (*High-idiomatic* Corpus and *Low-idiomatic* Corpus, respectively), used in the experiments described in Chapter 2 (see Section 2.3.1), for an *out-of-domain* comparison.

Datasets and setup for English/German. As our idiom dataset

for the English/German language pair (in both directions), we used the dataset of Fritzinger et al. (2010). This dataset was extracted from EUROPARL and from a German newspaper corpus[6]. The dataset consists of sentences containing one of 77 German preposition+noun+verb triples. Each sentence is annotated with syntactic and morphological features and has one of four labels: *idiomatic*; *literal*; *ambiguous*; or *error* (which indicates parsing or extraction errors). Around 95% of the dataset consists of idioms and the remaining 5% consists of the other 3 labels. Therefore, we considered only the 3,050 sentences extracted from EUROPARL given that it is a bilingual corpus. From these sentences, we randomly selected 2,200 sentences to build a test set and used the remaining 850 sentences as part of our training data so as to ensure that there were idioms in the training data.

The dataset of Fritzinger et al. (2010) contains only German sentences and, thus, we had to extract the English translations from EUROPARL. To do this we first searched the English/German EUROPARL corpus for the German sentences matching those from the idioms dataset using the simple heuristic *is equal or not*.[7] We then recorded the position of the relevant sentences in the German portion of the corpus and extracted the sentences in the same position from the English portion. To verify this translation extraction process we asked a native German

[6] *Frankfurter Allgemeine Zeitung*

[7] We were able to match all the 2,200 sentences in the test set as well the 850 in the training set for German using this approach.

Table 6.3
Results in terms of BLEU and TER score on the idioms test set extracted from the Fapesp-v2 test sets in both directions. "Dot" and "General" refer to the content function applied to compute the attention over the encoder states. Larger BLEU and smaller TER scores indicates better performance.

Model	Idioms in Fapesp-v2	
	BLEU	TER
English/Brazilian-Portuguese (EN/PT-BR)		
NMT	32.5	47.9
SMT	**38.9**	**60.6**
Dot & *Attentive* NMT w/ *single* score function	32.7	47.1
Dot & *Attentive* NMT w/ *combined* score function	35.5	47.6
General & *Attentive* NMT w/ *single* score function	17.7	65.7
General & *Attentive* NMT w/ *combined* score function	15.8	65.6
Brazilian-Portuguese/English (PT-BR/EN)		
NMT	33.8	45.3
SMT	**47.4**	**34.3**
Dot & *Attentive* NMT w/ *single* score function	35.4	44.6
Dot & *Attentive* NMT w/ *combined* score function	34.1	47.0
General & *Attentive* NMT w/ *single* score function	18.8	63.4
General & *Attentive* NMT w/ *combined* score function	17.8	63.0

speaker who is also fluent in English to check a random selection of the extracted English/German sentence pairs. In the remainder of this paper we call the resulting dataset "German Idioms". We also included the WMT' 2014 and WMT' 2015 test sets in our evaluation to compare the performance of our systems on general language.

Given the fact that we have removed those idioms from the training corpora used for both our models and our baselines, we re-used all these models in this experiment (see Section 6.3).

Table 6.4
Results in terms of BLEU and TER score on the *High-idiomatic* Corpus and *Low-idiomatic* Corpus. "Dot" and "General" refer to the content function applied to compute the attention over the encoder states. "High" refers to the *High-idiomatic* Corpus. "Low idioms" refers to the *Low-idiomatic* Corpus. Larger BLEU and smaller TER scores indicates better performance.

Model	"High idioms"		"Low idioms"	
	BLEU	TER	BLEU	TER
English/Brazilian-Portuguese (EN/PT-BR)				
NMT	6.1	92.2	6.2	91.8
SMT	**19.8**	**61.4**	**21.1**	**60.1**
Dot & *Attentive* NMT w/ *single* score function	5.0	95.2	5.3	95.0
Dot & *Attentive* NMT w/ *combined* score function	5.2	96.9	5.4	96.9
General & *Attentive* NMT w/ *single* score function	5.1	95.0	5.2	95.3
General & *Attentive* NMT w/ *combined* score function	5.0	96.2	5.0	96.1
Brazilian-Portuguese/English (PT-BR/EN)				
NMT	5.8	92.7	6.1	95.1
SMT	**25.4**	**52.3**	**22.6**	**53.3**
Dot & *Attentive* NMT w/ *single* score function	4.8	103.6	4.9	101.8
Dot & *Attentive* NMT w/ *combined* score function	4.7	107.5	4.5	112.6
General & *Attentive* NMT w/ *single* score function	4.8	106.2	4.8	102.1
General & *Attentive* NMT w/ *combined* score function	5.0	110.0	4.6	112.1

6.4.1 Idioms Results

Table 6.3 presents the results in terms of BLEU and TER scores for the dataset containing idioms found in the Fapesp-v2 test sets. Although the *Attentive* NMT have shown improvements over the SMT baseline in both test sets of the Fapesp-v2 when literal language was considered, the performance was not maintained over the test set of *in-domain* idioms. The SMT baseline performed best in both directions in terms of both BLEU and TER, with an increase of more than 3 BLEU points in EN/PT-BR direction and more than 10 points in the PT-BR/EN direction.

The NMT baseline had a performance closer to our *Attentive* NMT

but still 3 BLEU points below our best system in both directions and only slightly below our systems in terms of TER, also in both directions. Although the NMT baseline performed worse than our models, the difference was smaller in comparison to the results obtained over the test sets of regular language.

However, the biggest change in the results come from the fact that our best model over regular language, *i.e.* the General & *Attentive* NMT with *single* score, had the worst performance overall in the test set of idioms in both Brazilian-Portuguese to English and from English to Brazilian-Portuguese. Its performance dropped by almost 20 points in terms of BLEU and also in terms of TER.

Table 6.4 presents the results in terms of BLEU and TER scores for the *High-idiomatic* Corpus and *Low-idiomatic* Corpus used in our experiments in Chapter 2. Surprisingly, there was a big drop in the performance of all NMT systems over these datasets. All NMT systems, including ours and the baseline, performed poorly on both datasets. The BLEU scores dropped around 30 BLEU points on average in comparison to the results achieved by the same models over regular language.

The SMT system maintained a good level of performance over these *out-of-domain* corpora achieving far more BLEU points than the NMT counterparts in both directions.

Table 6.5 presents the results in terms of BLEU and TER scores over the dataset of idioms of Fritzinger et al. (2010). Given that this idiom

Table 6.5
Results in terms of BLEU and TER score on the idioms test set for the English/German language pair in both directions. "Dot" and "General" refer to the content function applied to compute the attention over the encoder states. Larger BLEU and smaller TER scores indicates better performance.

Model	Idioms dataset	
	BLEU	TER
German/English (DE/EN)		
NMT	20.1	64.2
SMT	**24.0**	**60.6**
Dot & *Attentive* NMT w/ *single* score function	13.9	74.1
General & *Attentive* NMT w/ *single* score function	14.3	73.8
English/German (EN/DE)		
NMT	14.4	72.4
SMT	15.9	68.7
Dot & *Attentive* NMT w/ *single* score function	19.5	66.9
General & *Attentive* NMT w/ *single* score function	**19.9**	**64.1**

corpus was extracted from the data used to train all models, we can consider it an *in-domain* test set.

Surprisingly, in the English/German direction, both of our models had the worst performance among all models in terms of both BLEU and TER scores. Although the NMT had a worse performance over the regular language, it had a stronger performance over the test set of idioms. The SMT system performed best in this language direction, achieving almost 20 BLEU points more than our best model.

Nevertheless, in the English/German direction, both of our models performed better than all the baselines, including the SMT, in terms of both BLEU and TER. Our models had a similar performance when the BLEU scores are considered, but there was a difference of almost 3 TER

points between our best and our worst models.

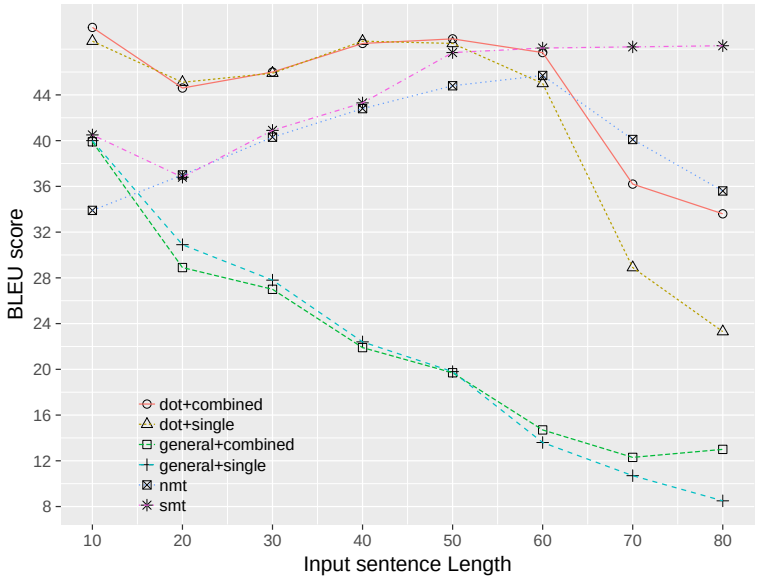
6.5 Analysis of the Models

In this section we will analyse the behaviour of the *Attentive* NMT models. As in Chapter 5, we split the analysis into a section about the behaviour on regular language (Section 6.5.1) and a section about the behaviour on idiomatic language (Section 6.5.2). We provide a discussion on our findings in Section 6.5.3.

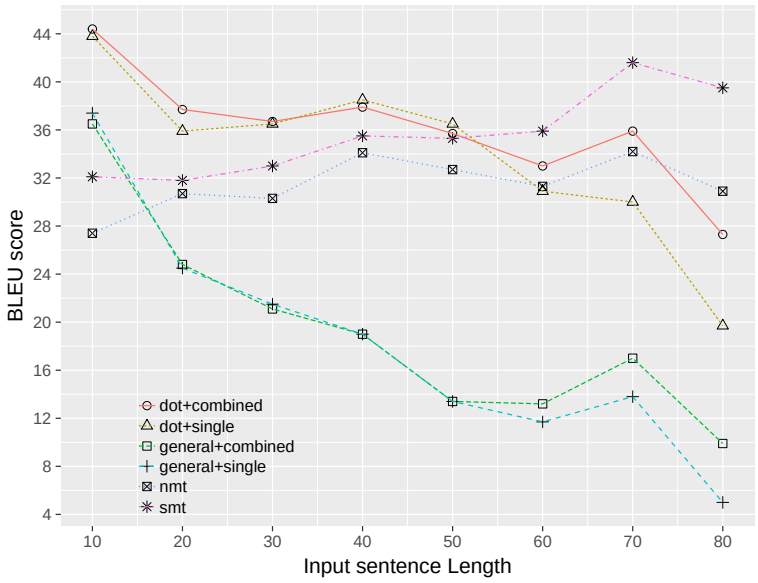
6.5.1 Regular Language

Figure 6.4 show a plot of the BLEU scores obtained by the systems for different sentence lengths over the test-a and test-b corpora of Fapesp-v2 in the English/Brazilian-Portuguese direction. As we can see from that figure, our worst systems (both *Attentive* NMT with *general* encoder content function) in both directions of the English/Brazilian-Portuguese language pair had a similar performance when a split by sentence length is considered. Both systems started with a high BLEU score for sentences up to 10 words but that performance rapidly degraded as the sentence length was increased.

Our two best systems (both *Attentive* NMT with *dot* encoder content function), maintained a performance above both baselines) up to sentences 50 words long. Coincidentally, this is the same maximum size



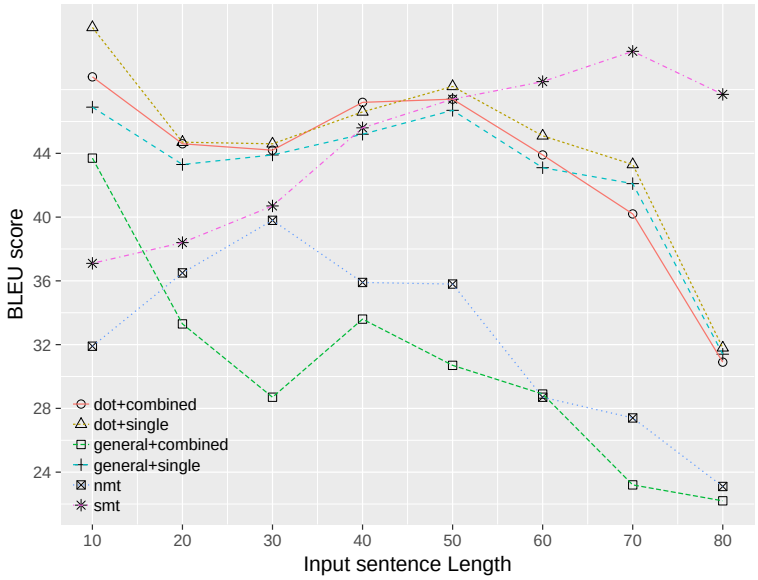
(a) Fapesp-v2 test-a



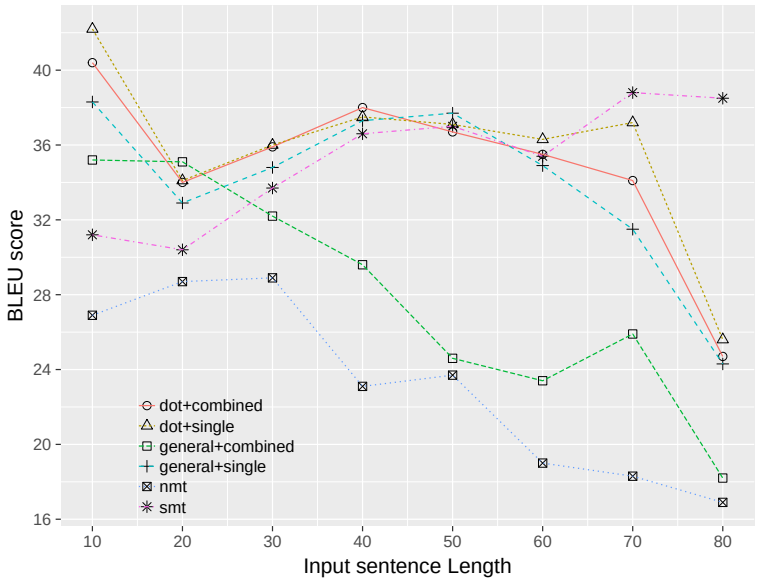
(b) Fapesp-v2 test-b

Figure 6.4
BLEU scores (EN/PT)

for sentences used during the training step of all our models. However, in this particular language pair/test sets, the SMT system does not to have its performance degraded by sentence length. In fact, the SMT baseline had its best BLEU scores on sentence of 60 words and longer,



(a) Fapesp-v2 test-a



(b) Fapesp-v2 test-b

Figure 6.5
BLEU scores (PT/EN)

performing better than our best systems and the NMT baseline.

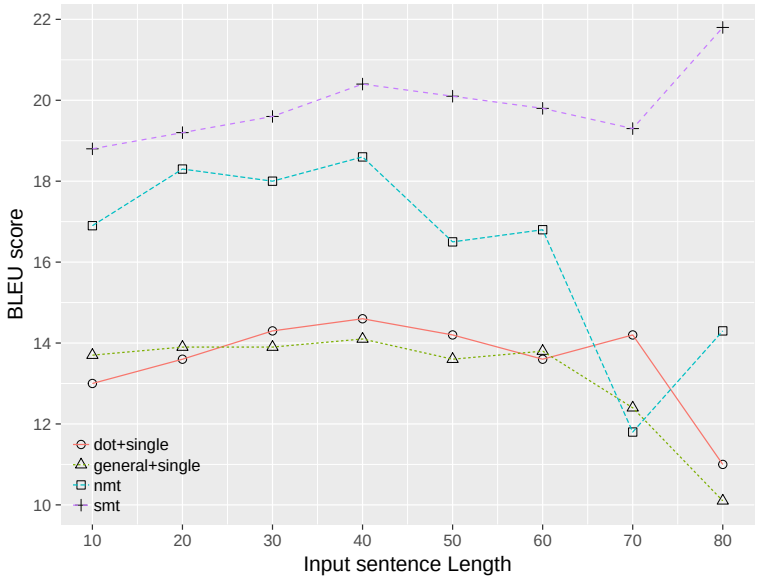
The NMT baseline, although having performed consistently worse than the SMT baseline and both of our best models and had a stronger performance than our second best model for sentences of 70 words and

longer. Furthermore, the NMT baseline had a stronger performance than our best model for sentences longer than 80 words.

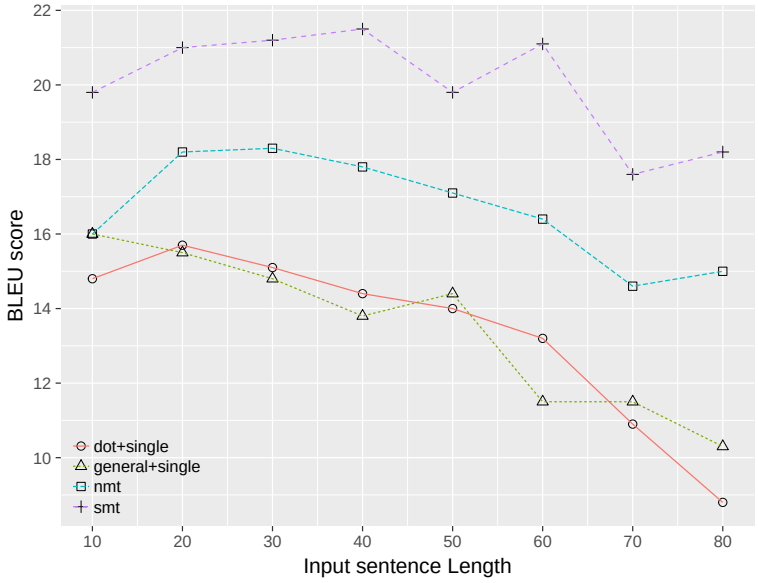
Figure 6.5 show a plot of the BLEU scores obtained by the systems for different sentence lengths over the Fapesp-v2 test corpora in the Brazilian-Portuguese/English direction. Some of the patterns in the systems' performance in the English/Brazilian-Portuguese direction are also present in the Brazilian-Portuguese/English direction. The *Attentive* NMT with *dot* encoder content function consistently performs better than the baselines on sentences up to 50 words long (the same maximum length used for training). For sentences longer than 50 words, the SMT once again improved its performance on the BLEU score and is not negatively affected by sentence length.

One of the differences in this language pair/direction in terms of BLEU scores is the increased performance of the *Attentive* NMT with *single* score and *general* encoder content function. Although it does not achieve the best results in this language pair, their performance was similar to the other *Attentive* NMT with *combined* score and *dot* encoder content function which were the best models in terms of BLEU score.

Another difference in this language pair/direction, is the drop on the baseline NMT performance. The baseline NMT had difficulties on dealing with sentences above 30 words in length. Although it had a strong performance, similar to the SMT model, in the English/Brazilian-Portuguese direction, that performance was not observed in the opposite



(a) WMT' 2014



(b) WMT' 2015

Figure 6.6
BLEU scores (DE/EN)

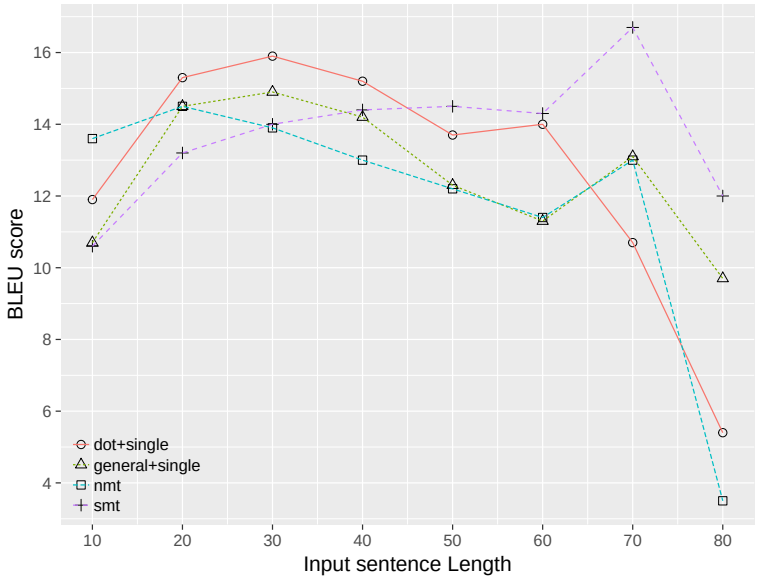
direction.

Figure 6.6 shows a plot of the BLEU scores obtained by the systems for different sentence lengths over the WMT' 2014 and WMT' 2015 test corpora in the English/German direction. Once again the SMT systems

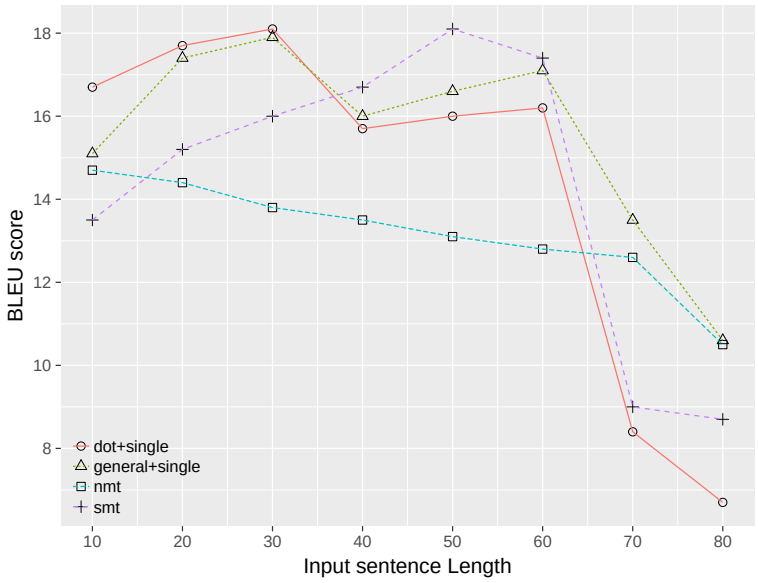
achieve a superior performance over longer sentences, although for this language pair it was the best system among all different lengths. The NMT baseline performed the second best in both test corpora and we can see from the figure that the performance starts to degrade on sentences longer than 40 words in the case of WMT' 2014 and as soon as 20 words on the WMT' 2015.

Despite the fact that both of our *Attentive* NMT with *dot* encoder content functions maintained a similar performance among sentences up to 50 words in length, the performance degrades as in the English/Brazilian-Portuguese language pair as the sentence length is beyond the length used in training. In addition, although that performance kept almost constant, it was far below the baselines. On the WMT' 2015 data, the performance of our models degraded as soon as we move to sentences longer than 10 words.

In Figure 6.7 we show the BLEU scores obtained over the WMT' 2014 and WMT' 2015 corpora for all the systems in the German/English direction. In these results, we can see from the figure that the *Attentive* NMT with *single* score and *dot* encoder content function perform better than all the baselines for sentences up to 40 words in the WMT' 2014 corpus and up to 30 words in WMT' 2015 corpus. Although the SMT model outperforms our *Attentive* NMT models on sentences longer than 50 words in WMT' 2014 corpus, the difference is not as large as in the other previous experiments. In addition, Dot & *Attentive* NMT with *sin-*



(a) WMT' 2014



(b) WMT' 2015

Figure 6.7
BLEU scores (EN/DE)

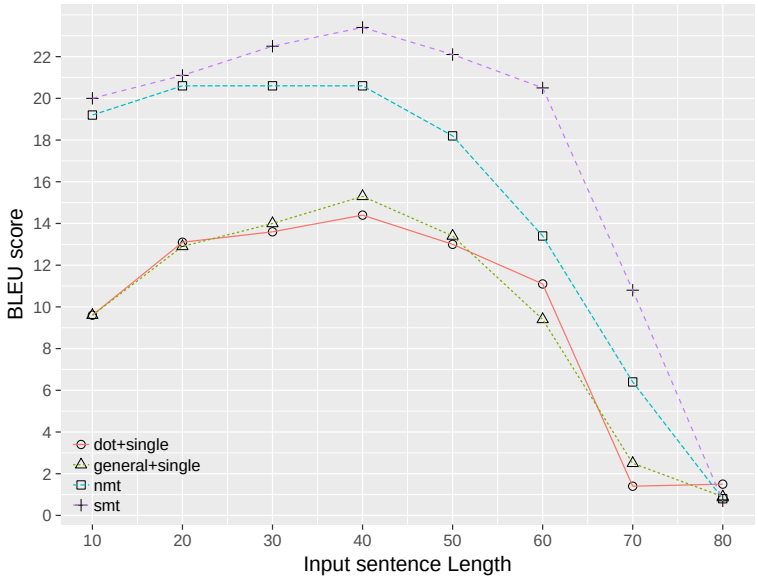
gle score was the best system for sentences up to 30 words in length but its performance degraded fast once the sentence length was increased to beyond 60 words in length. Although the General & *Attentive* NMT performed slightly below the other *Attentive* NMT for shorter sentences, it

performed better on longer sentences, although its performance started to degrade when the sentence of the length was 60 words or larger. Interestingly, the SMT baseline achieved larger BLEU scores for sentences between 40 and 60 words long, but its performance dropped below our best *Attentive* NMT and the NMT baseline.

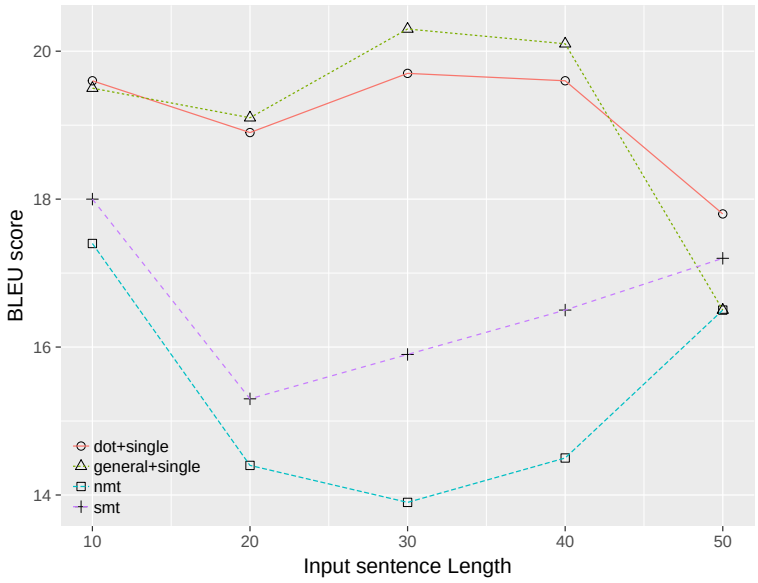
6.5.2 Idiomatic Language

Having analysed the performance of the systems over the test corpora containing regular language, we now provide an analysis over the idioms corpora used in the experiment. However, given the fact that the test set of idioms extracted from Fapesp-v2 were a small sample we could not make the same analysis as we did with literal language. In fact, the majority of the sentence lengths were in the range 20 to 40, with only 3 sentences outside that range (one shorter than 20 and two longer than 40). In addition, as we explained in Section 2.3.1, the *High-idiomatic* and *Low-idiomatic* corpora contained sentences with lengths in the range of $[15, 20]$ words and therefore, an analysis by sentence length is not indicative of any patterns.

Nevertheless, Figure 6.8 show the performance of the systems for the English/German language pair tested over the corpus of idioms of Fritzinger et al. (2010) in both directions. Once again, we display the BLEU scores obtained by the our and the baseline systems over sentences of different lengths.



(a) DE/EN



(b) EN/DE

Figure 6.8
BLEU scores on the idioms corpus

It is interesting to observe how a similar pattern of performance occurs in the DE/EN direction. All the systems improve their performance as the length of sentences increased up to 40 words and, in addition, all the systems also had a degradation in performance once the sentence

length went beyond 40 words. In addition, the SMT system, although the best system overall for this task, had the worst performance on the longest sentences of 80 words.

In the EN/DE direction, both the *Attentive* NMT had a superior performance on sentences up to 40 words in length. Once past that sentence length, there is a rapid degradation in the performance of both *Attentive* NMT systems although the SMT and NMT baselines increased their performance on the longer sentences. In addition, it is important to note that the lengths of the idiomatic sentences in the English input are in general shorter than their German counterparts after tokenization.

6.5.3 Discussion

The results obtained by our *Attentive* NMT are mixed. Although the models demonstrated a good performance in therms of BLEU score over the Brazilian-Portuguese/English language pair (in both directions) for regular language, the models failed to achieve the same level of performance in the German/English direction. Nevertheless, as shown by Jean et al. (2015b), NMT models often struggle when translating into English (in comparison to translating into other languages) and even the inclusion of an attention module to score past states of the decoder seems to work only for limited *in-domain* datasets, as the case of the Fapesp-v2 corpus.

Despite the fact that the *Attentive* NMT models do not achieve state-

of-the-art results apart from the English/Brazilian-Portuguese language pair, our *Attentive* NMT is on par with strong baselines composed of single models, including the state-of-the-art NMT model for the English/German language pair. In fact, our models consistently perform better than those NMT models in almost every sentence length in both language pairs analysed in terms of BLEU scores. The *Attentive* NMT, in general, perform well on sentences up to 50 words of length but its performance degrades very rapidly once that threshold is surpassed. The only exception to this behaviour is the English/German direction in which the *Attentive* NMT systems do not perform well on any sentence length.

When translating different languages, there are interactions between input and output words that are highly informative to the system when making the next prediction and that are captured via word alignments (Koehn, 2010). By introducing a module to score states of the decoder and subsequently merge this information to the alignments calculated by the attention module over the encoder states (as we do in Eq. 6.11) we are introducing a bias towards the decoder states and, thus, we may be weakening the information carried about input/output alignments. If the model is not robust enough to balance this trade off, it will fail and produce poor translations, a fact that is observed in several of our models. For example, all of our *Attentive* NMT models with a *general* encoder content function failed to converge for the English/German language

pair in all experiments and most of our systems failed to achieve the same level of performance as the baselines.

Nevertheless, although we recognize the introduction of the bias, we also show that the *Attentive* NMT models improved the translations from English into German, including the translation of idioms. Given the fact that the German language has long-distance dependencies in its syntax, this result may be an indication that our model helps in bridging those dependencies by bringing forward past information that may have faded from context at the target side with the *smoothing effect* (see Section 5.5). However, that improvement is not observed when English is the target side, which also indicates that the introduced bias can hurt the performance of the models.

Another point of consideration is the fact that the *Attentive* NMT models improved translation for the English/Brazilian-Portuguese language pair, in terms of BLEU scores, even though both languages in this pair may present shorter dependencies than German. In addition, the performance of the *Attentive* NMT over regular language was not carried over to the idioms dataset extracted from the same type of language register. By keeping only the idioms that can occur in the language domain of Fapesp-v2, the performance of the system degraded. Moreover, the same behaviour was observed in the baseline NMT.

The poor performance of our *Attentive* NMT and the baseline NMT models on the *High-idiomatic* and *Low-idiomatic* corpora in the En-

glish/Brazilian-Portuguese language pair, while performing well in the regular language, is an indication that these systems still have difficulties to translate *out-of-domain* text. In fact, these findings are in line with prior work (*e.g.*, Freitag and Al-Onaizan (2016) and Koehn and Knowles (2017)), that have shown that using a a system trained in one domain to translate text from other domains is still a challenge for NMT systems even if that model that can bring forward information from past contexts.

6.6 Conclusions

In this chapter we have studied the inclusion of the attention module over the decoder states of a NMT system. Although this type of attention module has achieved good performance in language modeling, such improvement were limited in the NMT setting. We have shown that an *Attentive* NMT system achieves state-of-the-art results for the English/Brazilian-Portuguese language pair even though it was trained on a small corpora. We also have shown that although our systems do not achieve state-of-the-art results for German/English direction, they are on par with single models that compose the ensembles that won the WMT' 2016 shared task.

Although the results are mixed, we have demonstrated that in some cases the representations built by the attention module also improves

the translation of idioms, especially when the target language has long-distance dependencies such as the case for German.

Chapter 7

Conclusions

Idioms are complex language constructions that exhibit semantic, syntactic and statistical idiosyncrasies. Previous approaches used to process idioms in NLP modeled those expressions in terms of their statistical properties and the context they are inserted in. Although achieving sometimes improved performance, more recent models explored simple representations for idioms while also increasing the amount of discourse history considered to build the representations.

In this thesis we have investigated the use of more complex representations for idiom processing across a range of tasks: *idiom type* and *idiom token* identification, language modeling and NMT. We have demonstrated that by using distributed representations the performance of models can be increased while greatly reducing the amount of discourse history that must be processed in order to achieve a good level of performance.

The reminder of this chapter present a summary of the contributions,

possible areas to expand our work, and our reflections and final remarks regarding the work presented in this thesis.

7.1 Summary of Contributions

A brief summary of the contributions made in this thesis are now presented:

i) In Chapter 2 we presented a quantification of the negative impact idioms have on the performance of SMT systems. We demonstrated that the performance of SMT baseline systems is greatly degraded when idioms are to be translated in comparison to regular language.

ii) We identified limitations in the current state-of-the-art in *idiom type* identification methods in Chapter 3 and proposed alternatives to overcome those limitations. We have shown that simple statistical representations can be used to represent local context about idioms but improved results are obtained through more complex representations.

iii) We demonstrated in Chapter 4 that by using distributed representations for sentences containing idioms the amount of discourse history needed to process such expressions can be reduced by a great extent. In addition, we have shown that by using these complex representations a single "general" classifier can be designed for *idiom*

token classification in stark contrast with the current approaches of training a different classifier for each expression.

iv) On further investigation into the use of distributed representations in Chapter 5 we have demonstrated that by including an attention module over the past internal states of an RNN-LM the performance of such models in terms of perplexity is greatly improved. In addition, we demonstrated that the attention module also improves the performance of the model for regular language, greatly reducing the size of the model (in terms of the number of parameters) to achieve results on par with larger state-of-the-art models. Moreover, we also demonstrated that the model is able to bridge long distant dependencies in language by recovering past information using the proposed attention module.

v) And finally, in Chapter 6 we adapted the attention module over the past internal states of the RNN-LM to a NMT system. Although we obtained a set of mixed results over regular language, we have shown that when translating idioms into languages with long-distance dependencies the attention module helps in bridging those dependencies. Moreover, we demonstrated the performance of the attention augmented NMT systems degrades on longer sentences in comparison with baseline NMT systems. In addition, we have shown that our model is also suited to languages with small,

good quality, *in-domain* bilingual corpora such as English/Brazilian-Portuguese.

7.2 Directions for Future Work

In this section we describe some areas of future research that we believe would be useful extensions to the work presented in this thesis.

Idiom type **identification**. A better analysis of the scores assigned to the VNIC pairs is needed, where one can distinguish the "hard" and "easy" cases for each fixedness metric and from this propose a better overall combination of these metrics other than the linear combination. We expect that this might produce better unsupervised approaches, specially to retrieve VNICs from corpora for under-resourced languages. We also believe that *deep learning* strategies might help to disentangle and capture the characteristics of idiom, given the idiosyncratic characteristics of idioms and supported by our results in both retrieval and classification tasks.

Idiom token **identification**. An investigation on the use of *Sent2vec* or other models that are able to encode larger samples of text such as Doc2Vec (Le and Mikolov, 2014) can provide insights about the shortest amount of context to maximize the performance of the "general" classifier. In addition, a further analysis of the errors made by the "general" model might highlight the types of expressions that the model is more

prone to make incorrect predictions for.

Distributed representations. Developing a better general understanding of distributed representations and what type of information these representations capture about language would help in the design of better models of idiom token identification. Indeed, and improved understanding of distributed representations would likely be useful in the design of a range of NLP systems, including language modeling and NMT.

Language Modeling. Tailoring the design of the attention mechanism in the language models to identify certain linguistic aspects such as part-of-speech or syntactic labels can aid the model when processing languages that have more flexible word ordering, such as Brazilian-Portuguese and French. In addition, investigations on how to span the attention to beyond sentence boundaries may help to bridge dependencies that are spread over, for example, a paragraph or an entire document.

Neural Machine Translation. The interactions between input and output language certainly are affected by the use of an attention mechanism over the states of the decoder RNN, and a quantification of that impact may shed light on why certain models fail to converge. Another potential area for investigation is how these models can be used for tree-to-tree or string-to-tree translations where syntax is directly involved in the process.

***Attentive* models**. Attentive models can be used in may NLP ap-

plications where long distant dependencies are involved. For example, the *Attentive* NMT model can be used to produce linearized versions of parse trees, in which the output can become very long when there are many branches in such tree.

7.3 Final Remarks

We have shown in this thesis that by using complex representations an idiom can greatly reduce the size of the context required to process such expressions. Although some of our results in some of the tasks are not state-of-the-art results, they are still on par with state-of-the-art systems while requiring much less information as input to do so.

Statistical representations in fact yield good performance in idiom related tasks. While it is not possible to capture all the nuances that permeate the usage and creation of idioms only with statistical representations, by framing idiom processing tasks in terms of the context in which these expressions are inserted, NLP systems can still perform reasonably well when using such representations. However, further improvements can only be achieved if a richer set of representations are in use in cases where the required amount of context is not accessible.

Even on tasks where context can be discarded, such as *idiom type* identification, the usage of high-dimensional representations (even when those originated from statistical analysis of the corpus are considered)

achieve remarkable improvements in the performance of the models. Moreover, the model is not only more efficient in terms of finding more *idiom types*, but also more confident when labeling a candidate expression as an idiom.

Idiomatic and literal meaning are completely different. While the latter retains the original meaning of every word, the former requires the words to become polysemous in its context. The improvements demonstrated by the use of distributed representations on tasks where the context is known beforehand, such as *idiom token* identification, are in line with findings on distributed representations for individual words and sentences. Those studies have shown that words and sentences with similar meaning tend to be close in feature space and, although idiomatic and literal meaning are not clear cut in some cases, these representations allow the model to identify to some extent when an input is idiomatic or literal, without having to retrain or fine-tune the classifier.

Many parallels were drawn on the distributed representations generated by machine learning models (such as RNNs) and the occurrence of distributed representations in the human brain. In fact, the most accepted psycholinguistic theory for idiom recognition by humans, the " Configuration Hypothesis", requires that such distributed representations are present in the human brain. Although we do not intend to claim that distributed representations do occur in human brains or to prove that the "Configuration Hypothesis" is correct, the fact that the representations

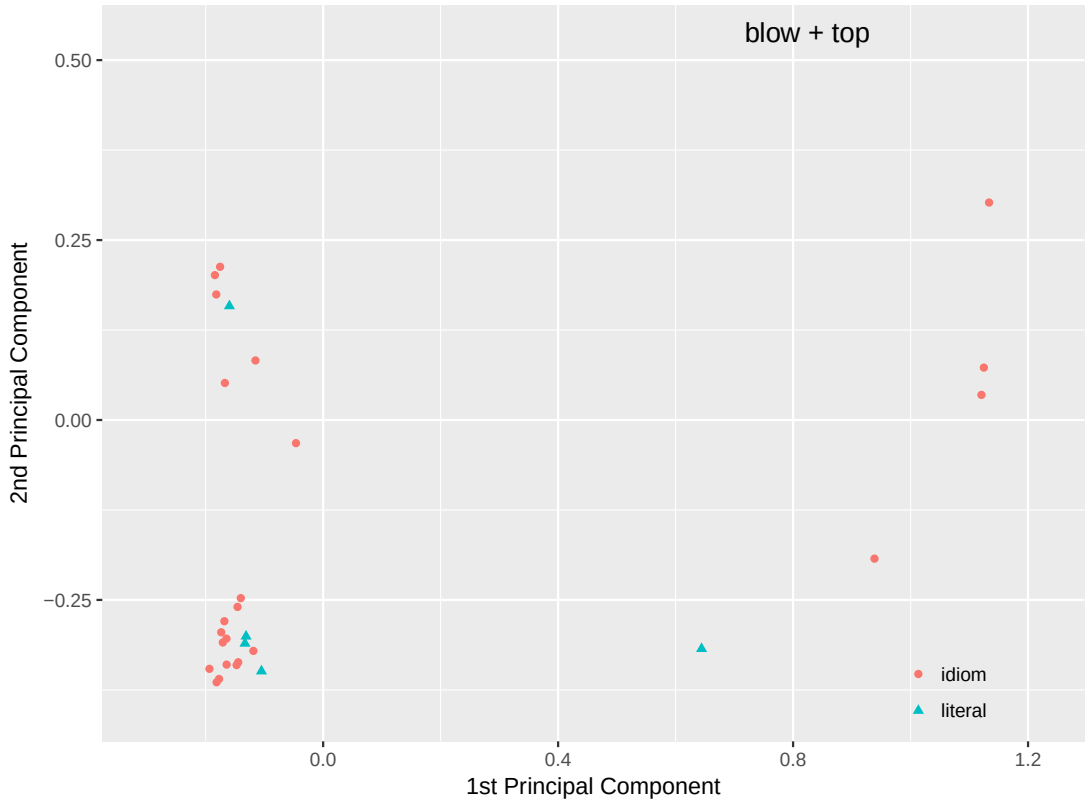
generated by RNN-based models improve idioms processing across a range of tasks is in line with psycholinguistic studies. Given that these studies often present as inputs to the participants only the sentence containing the candidate idiom only, we see our contributions as a another indication on the similarities between both types of distributed representations, in human brains and those generated by machine learning.

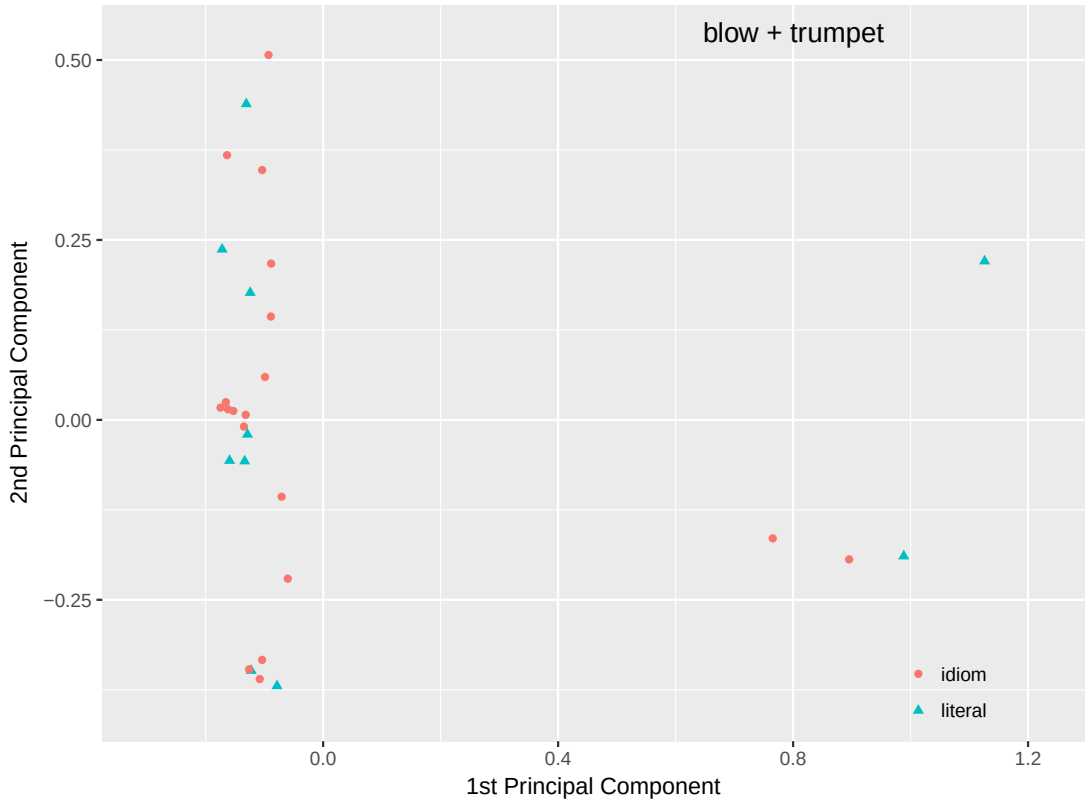
As is the case with psycholinguistic studies, it is difficult to interpret what distributed representations generated by RNNs are capturing from the data. In fact, as pointed in our future work (Section 7.2), a better understanding of such representations will guide the model design towards better models suited to idiom processing.

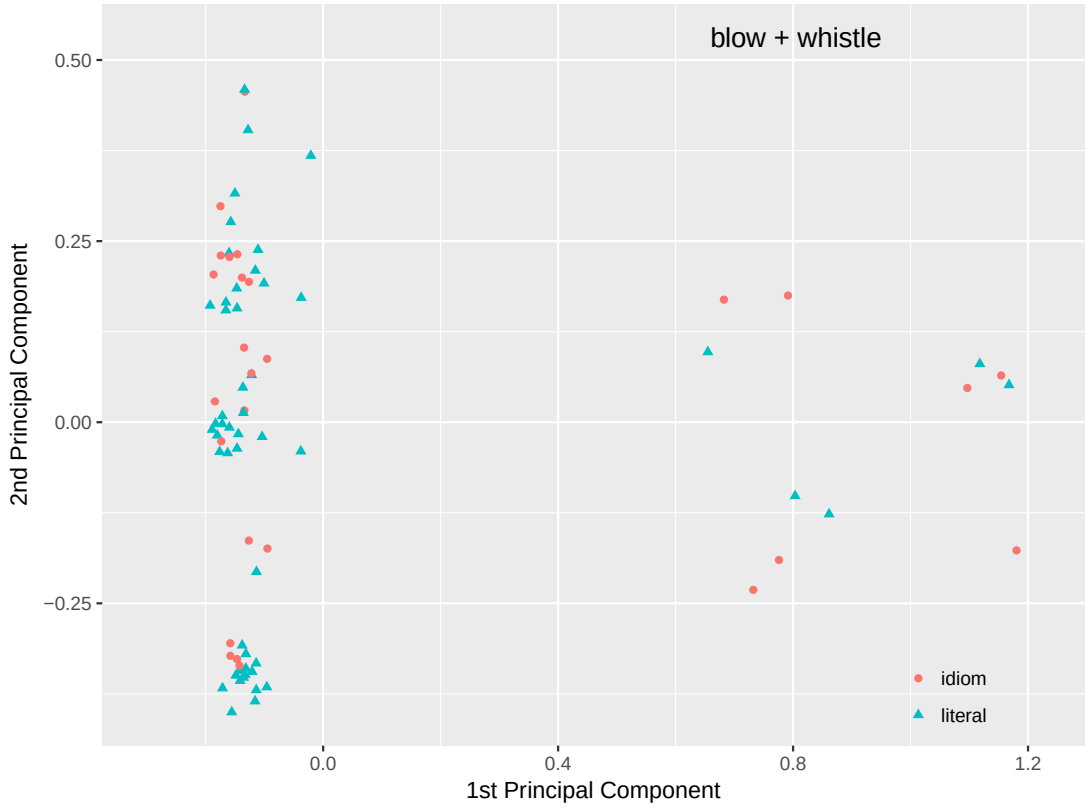
We believe that understanding the information carried by the distributed representations, researchers will be able to tailor their models towards capturing specific components of language and, consequently, result in better models for several different NLP tasks including, but not limited to, the computational processing of idiomatic language.

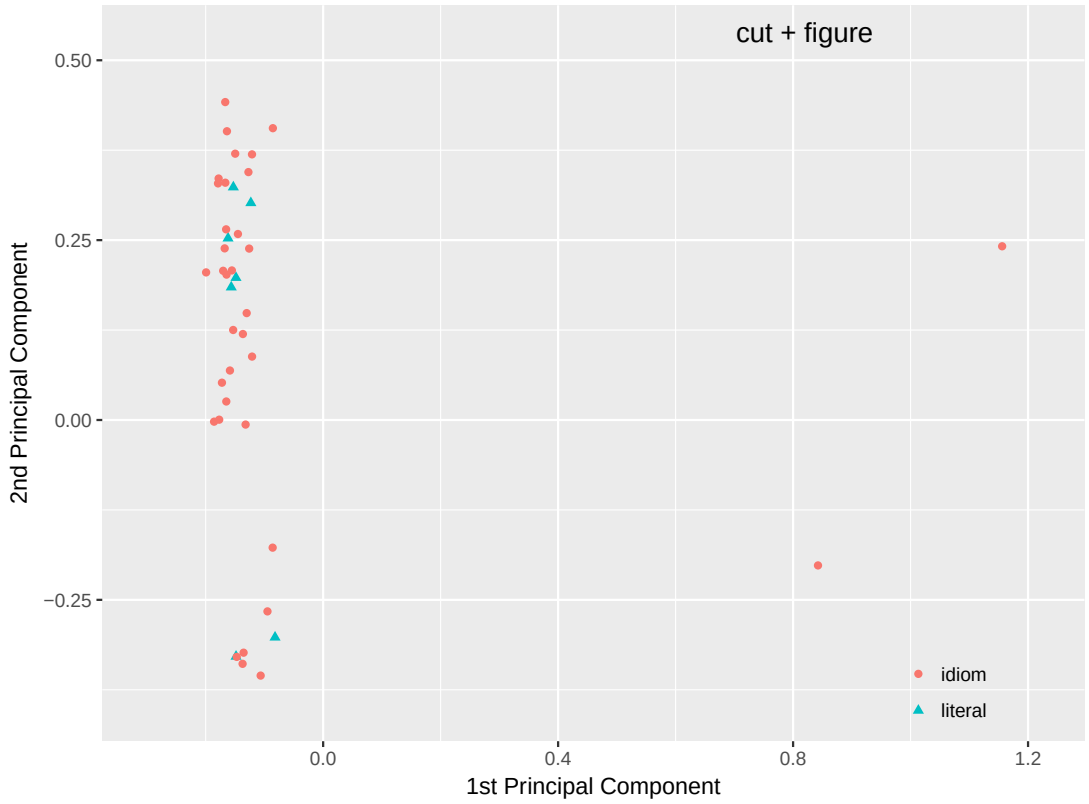
Appendix A

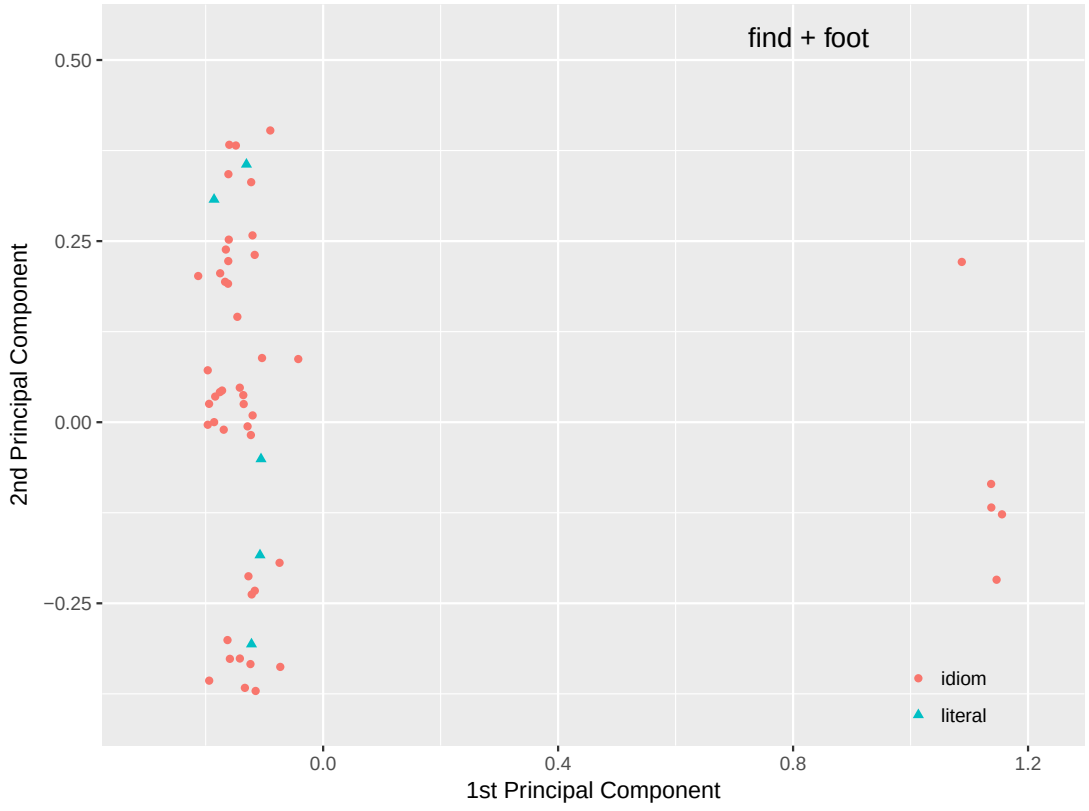
Individual Plots of Principal Components of Distributed Semantics

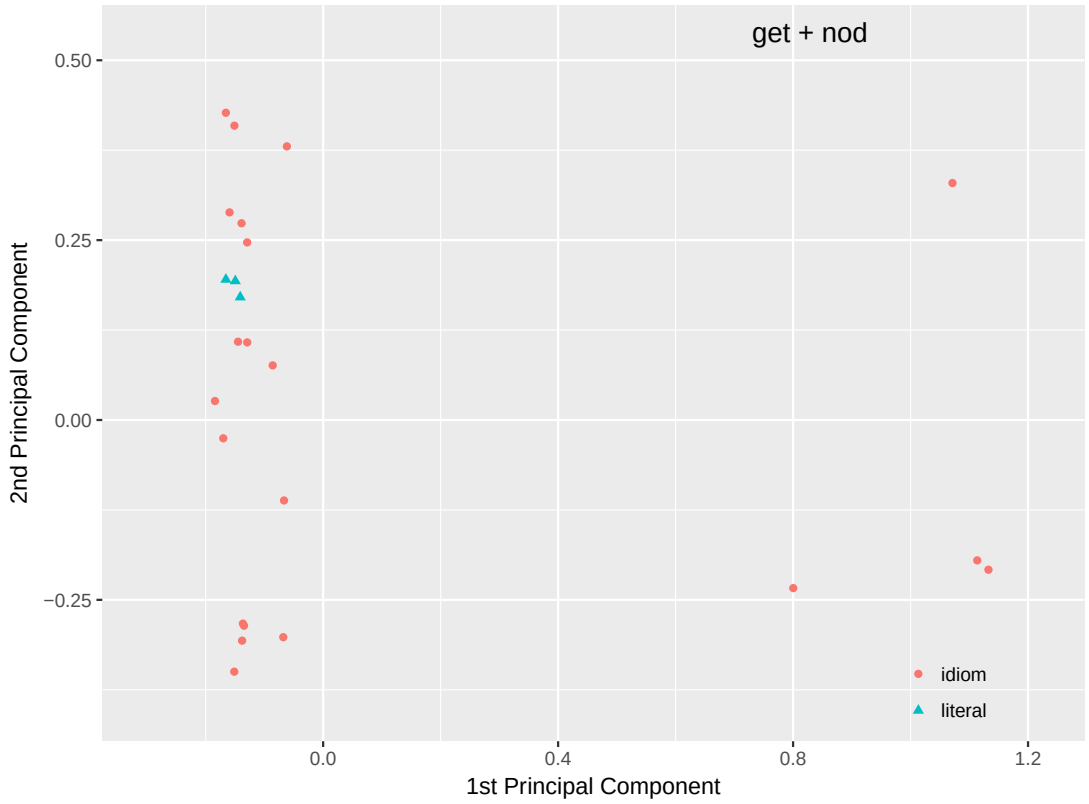


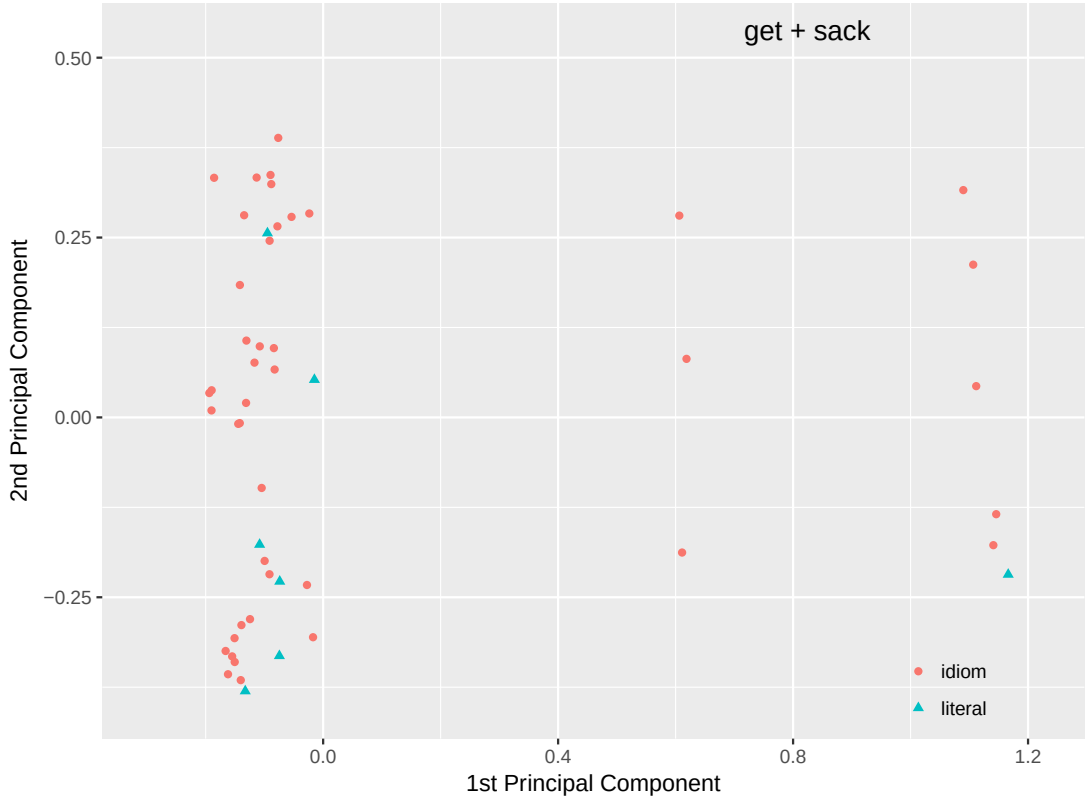

blow + trumpet
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

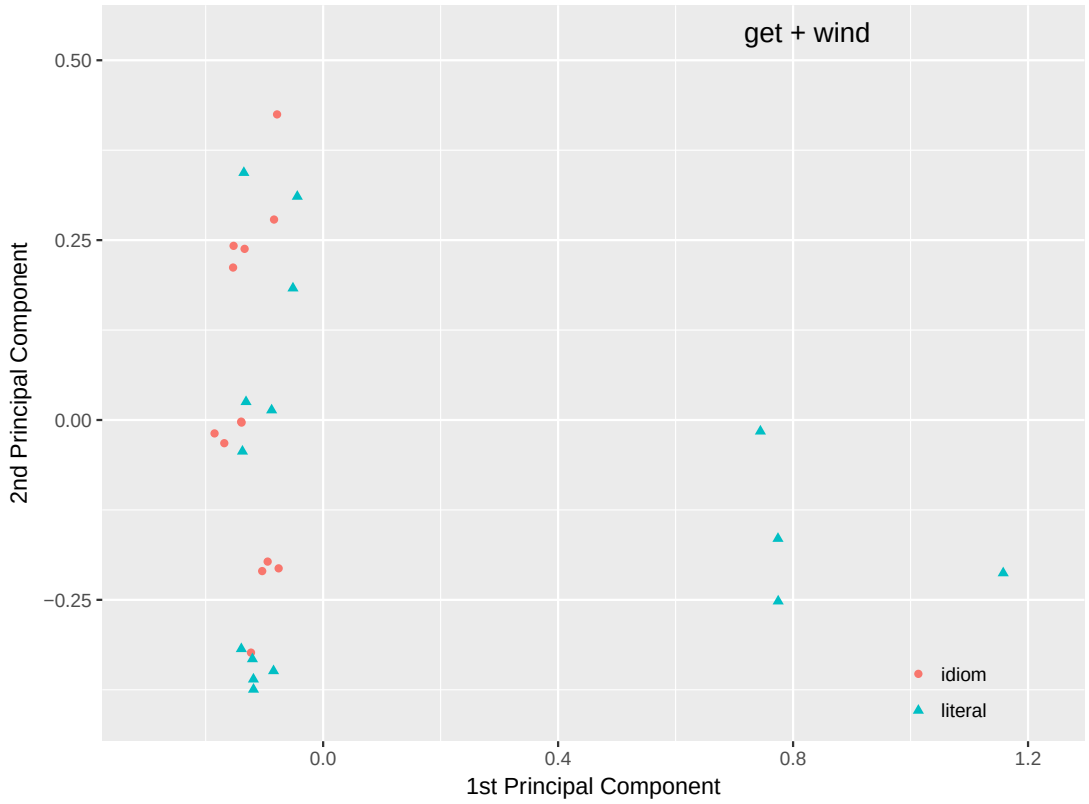

blow + whistle
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

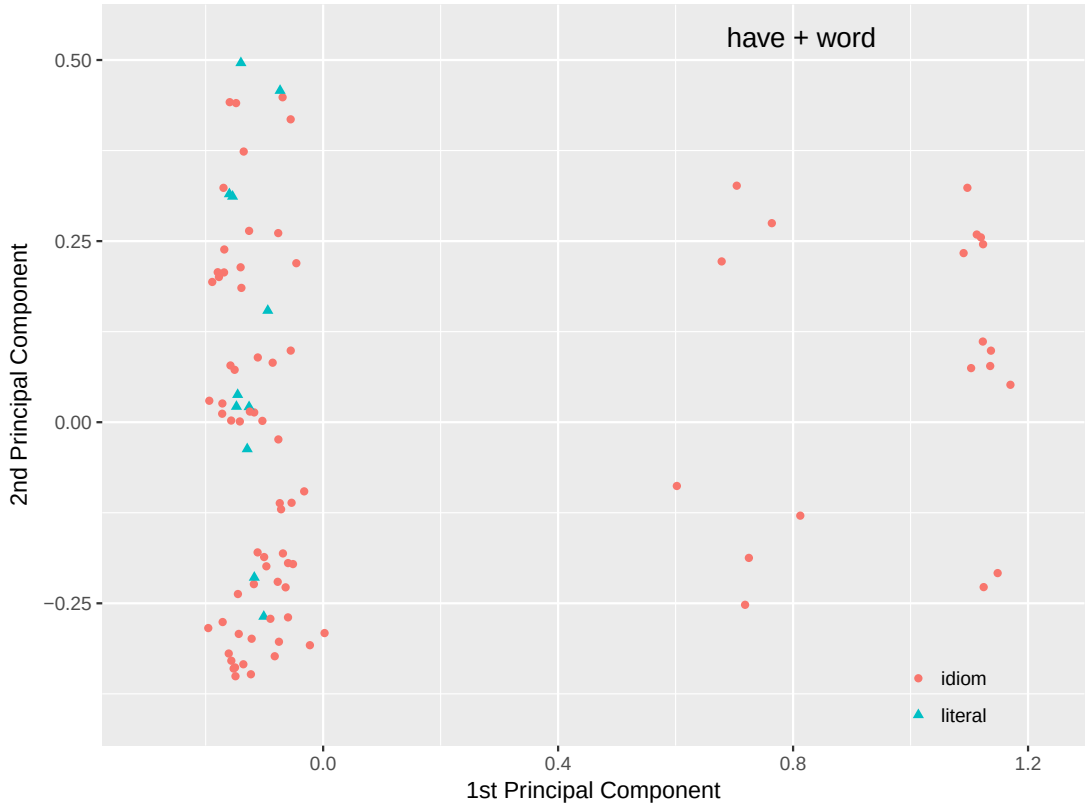

cut + figure
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

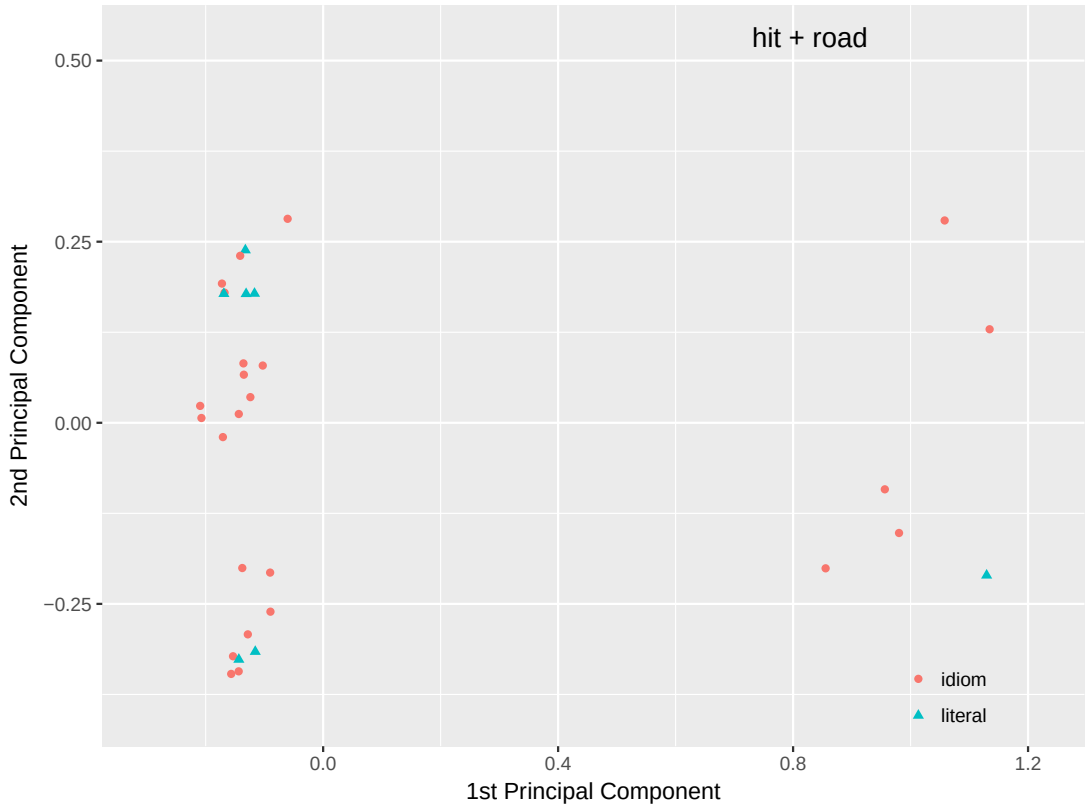

find + foot
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

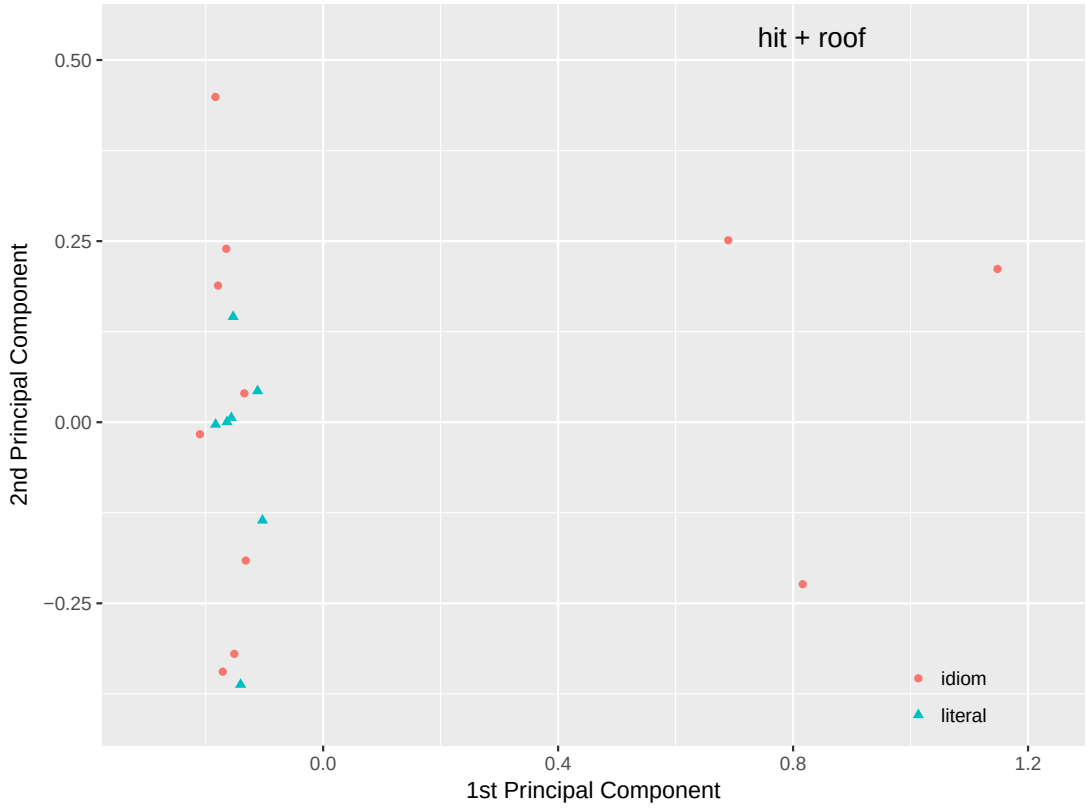

get + nod
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

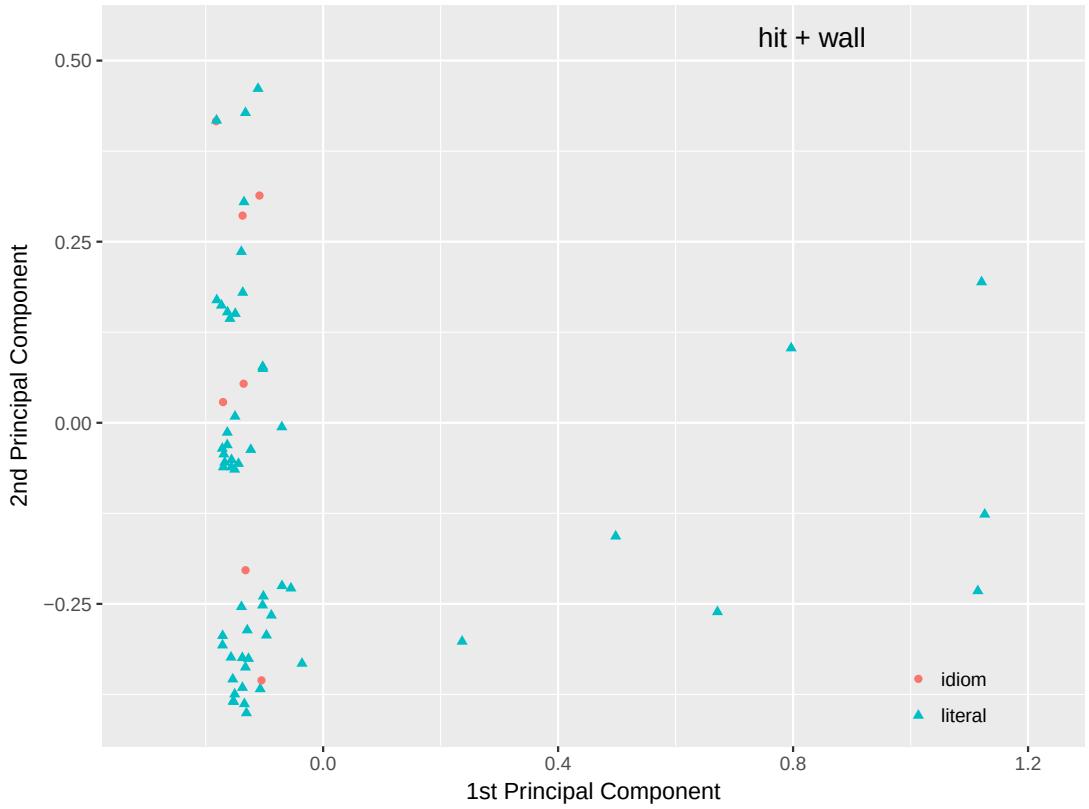

get + sack
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

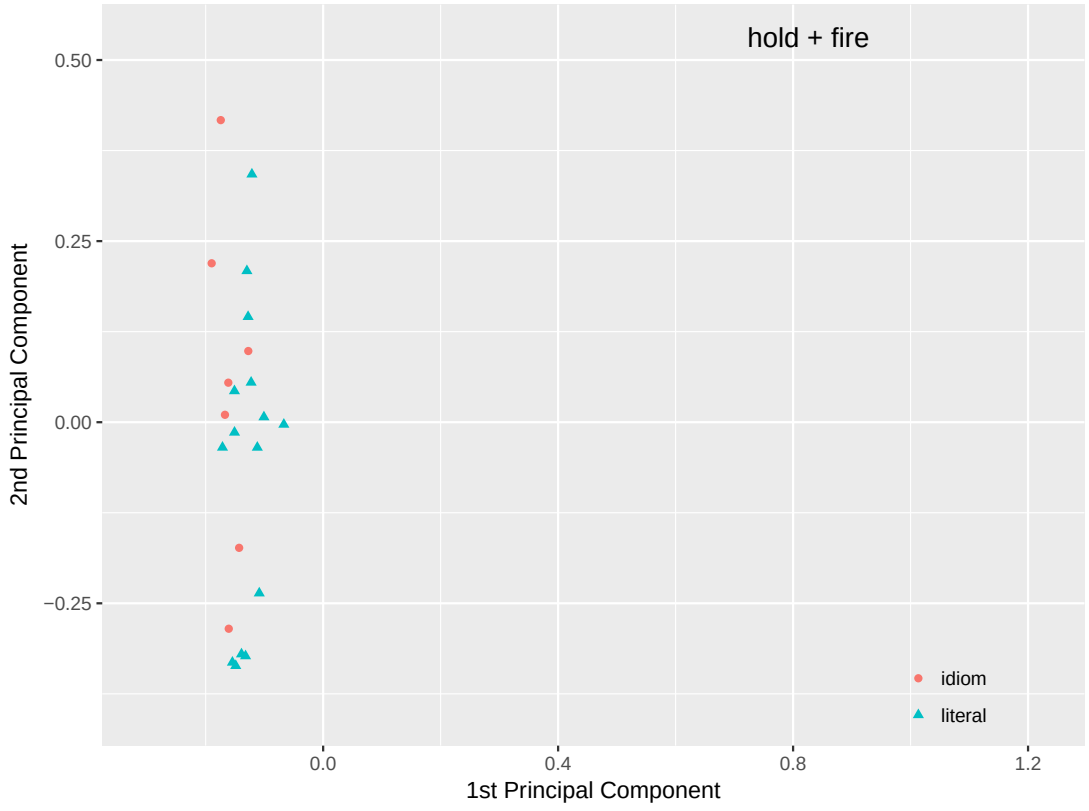

get + wind
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

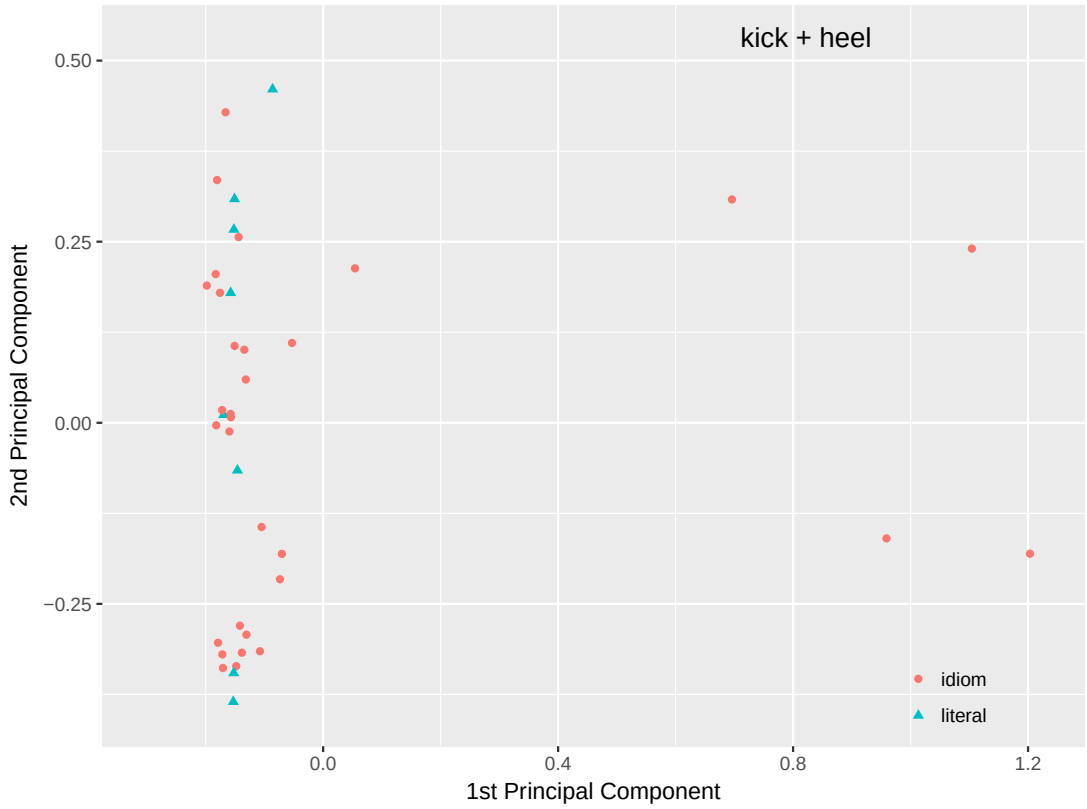

have + word
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

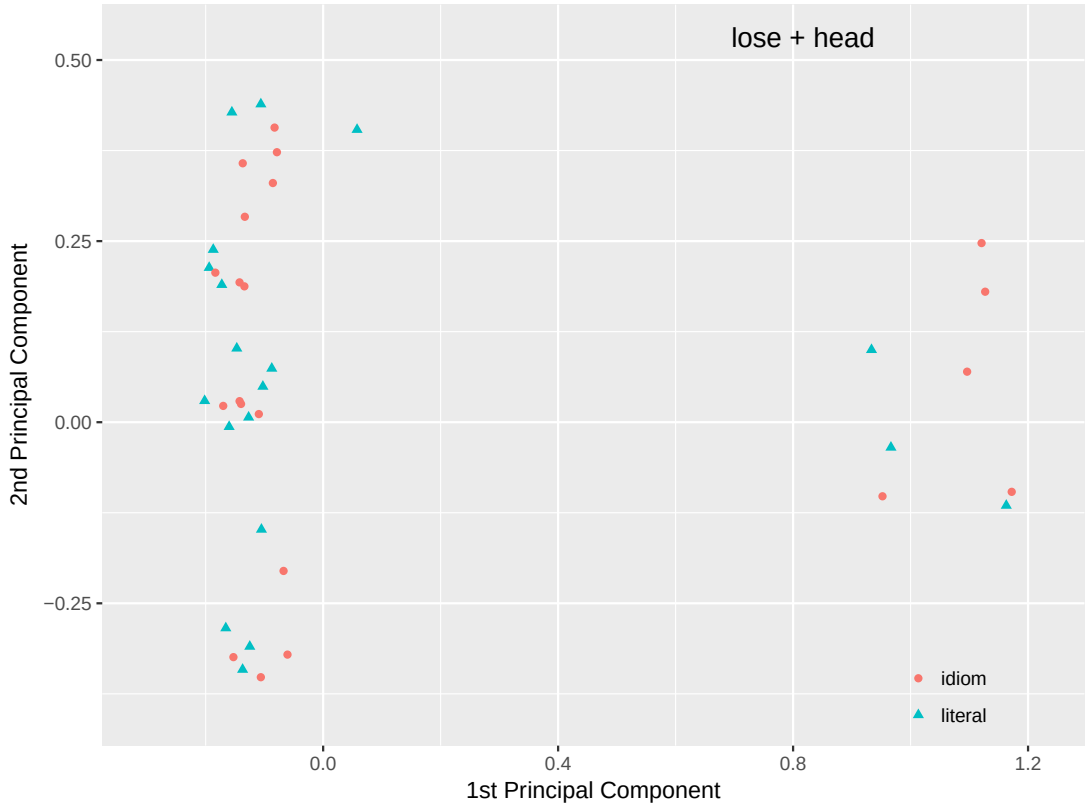

hit + road
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

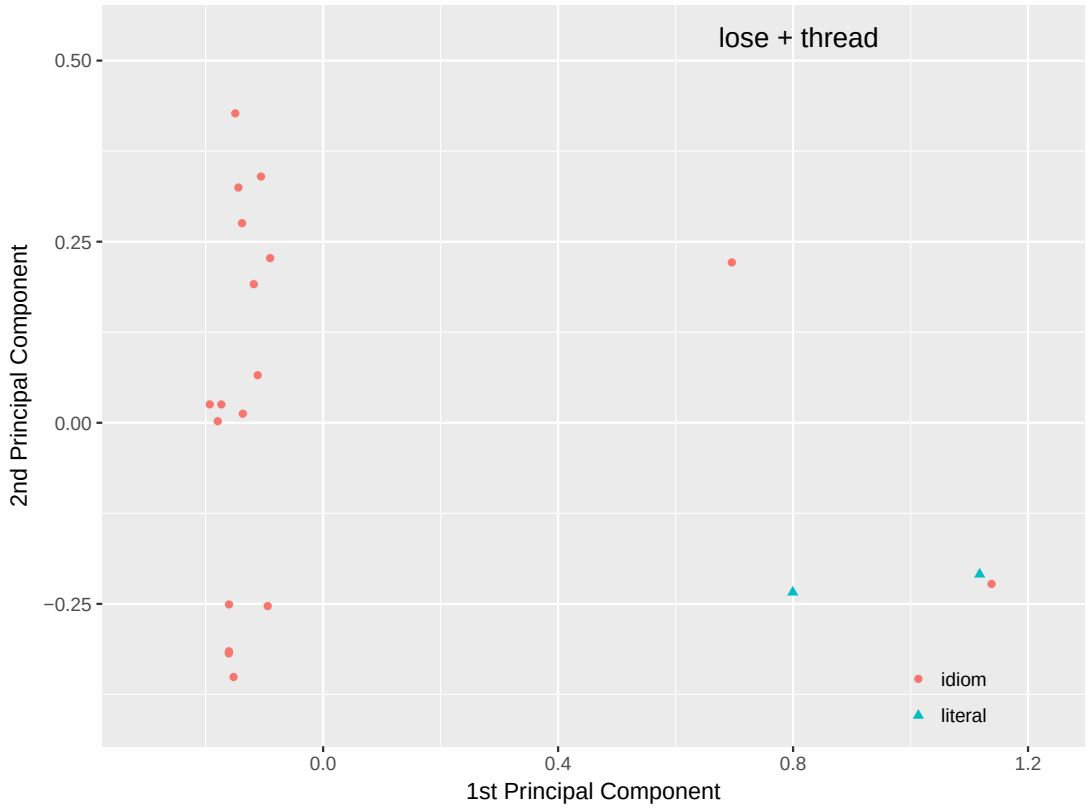

hit + roof
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

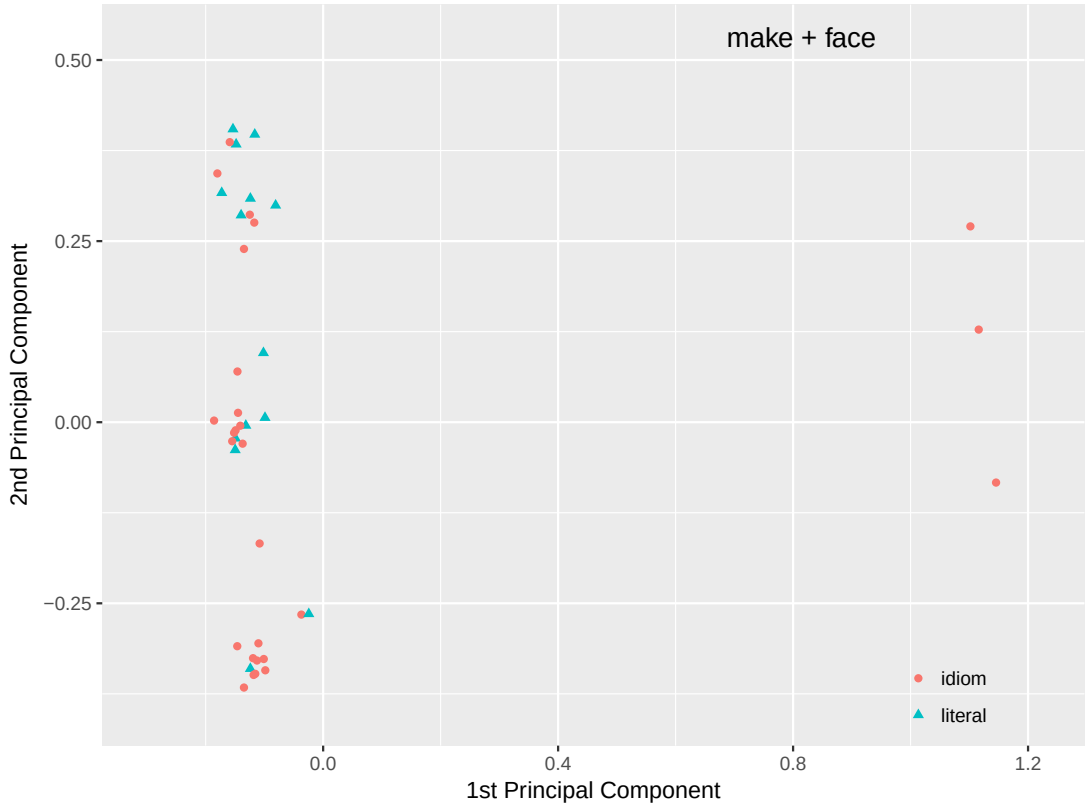

hit + wall
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

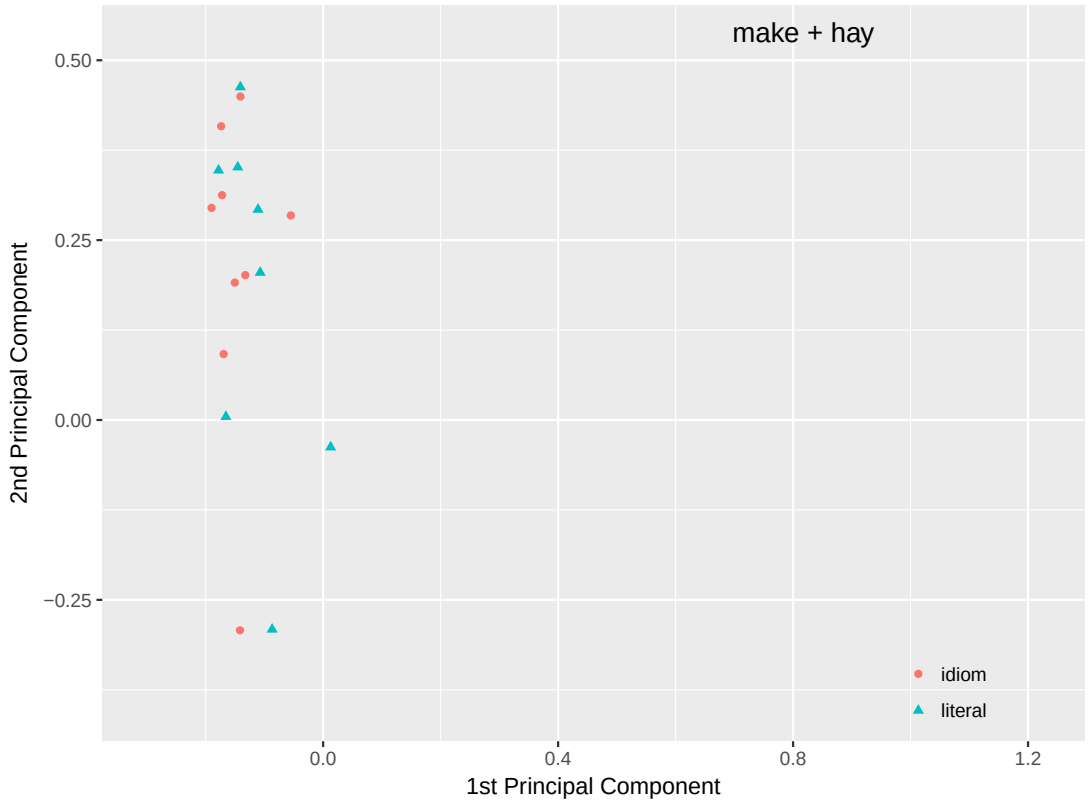

hold + fire
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

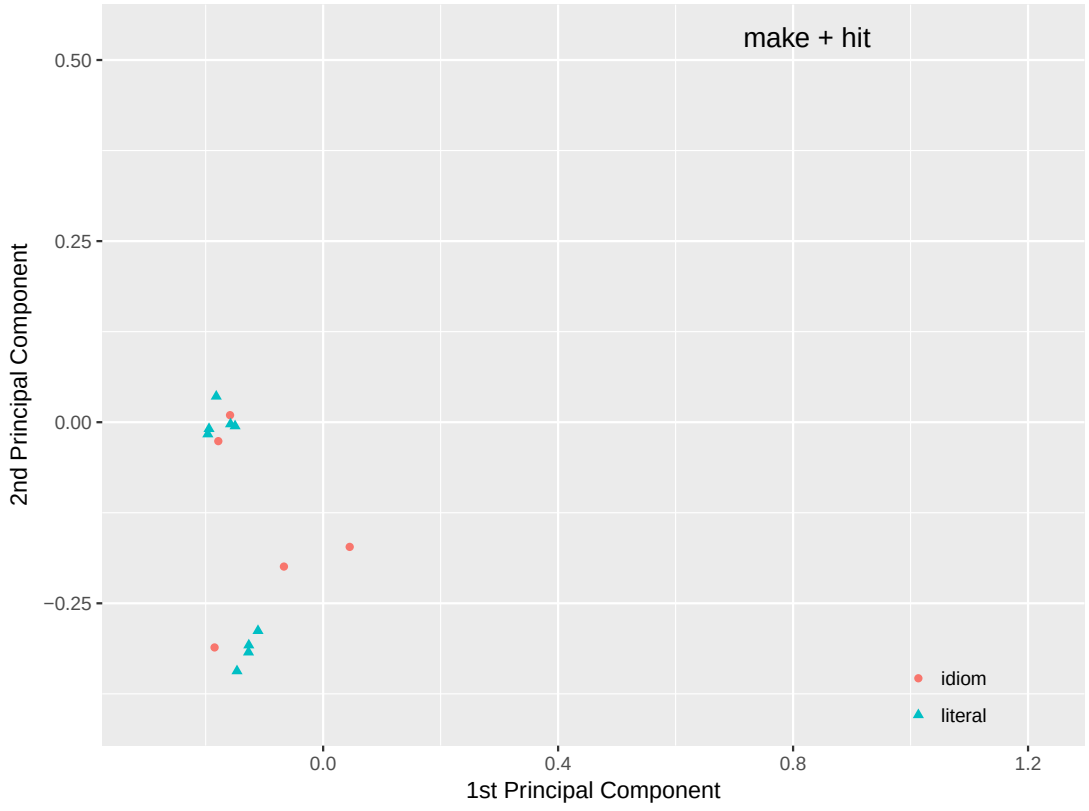

kick + heel
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

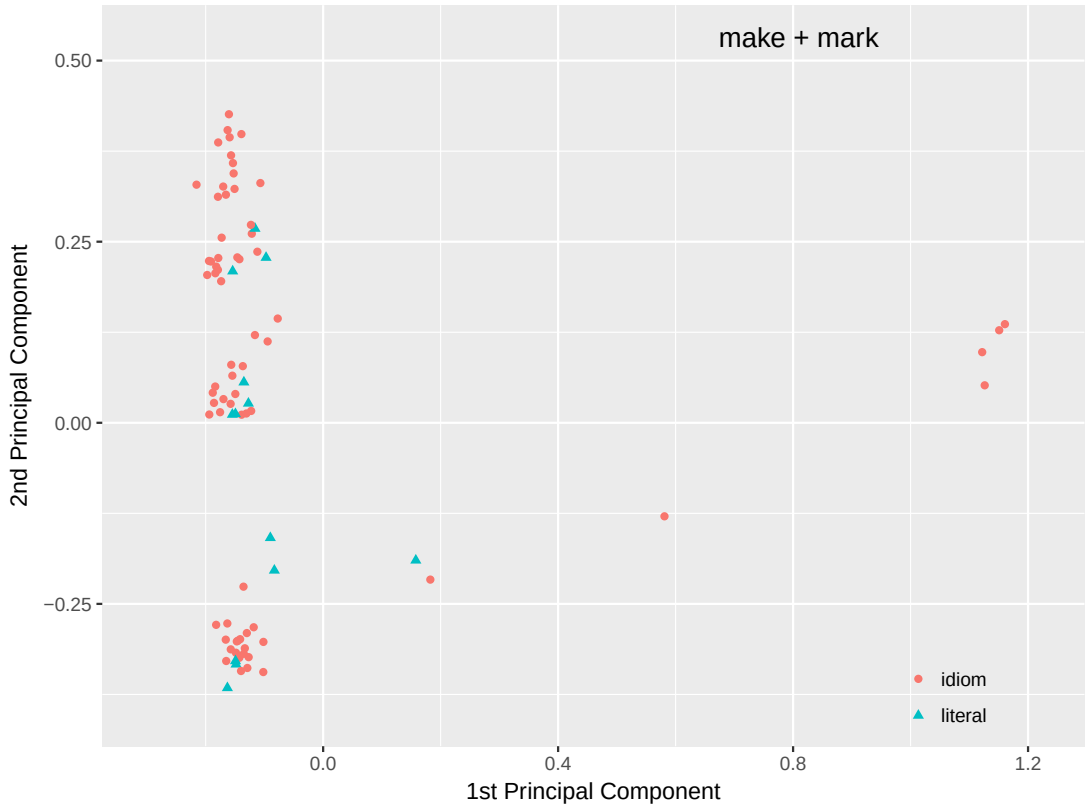

lose + head
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

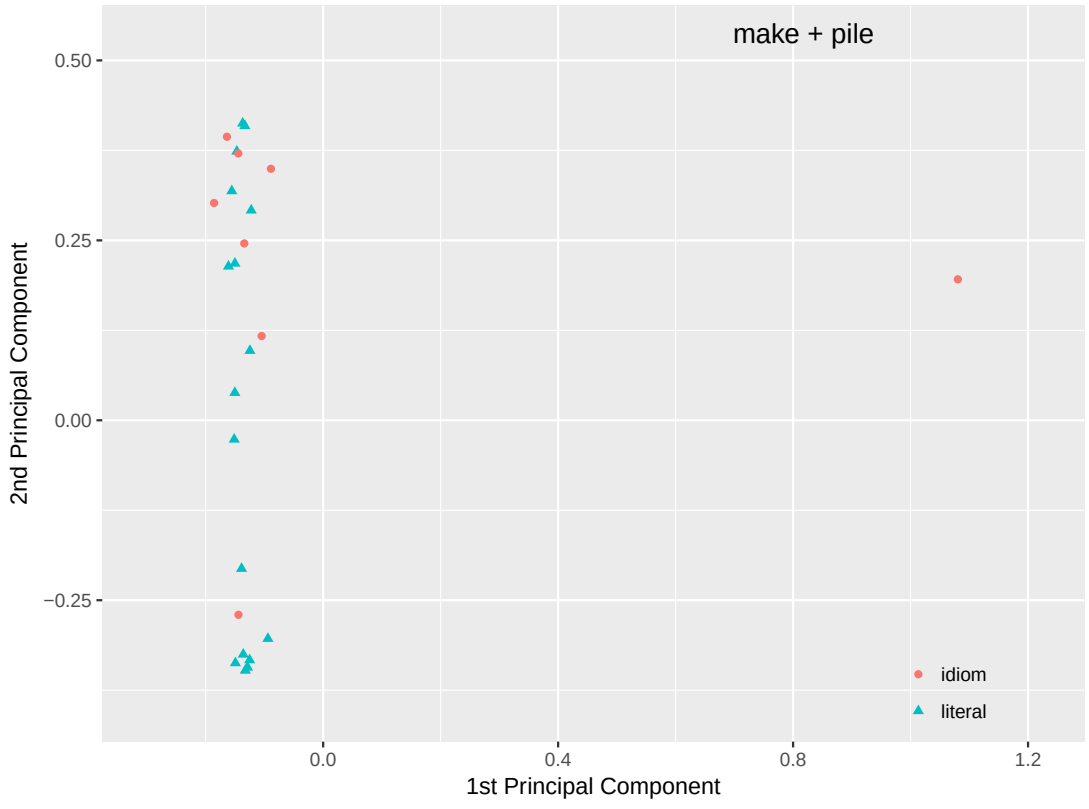

lose + thread
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

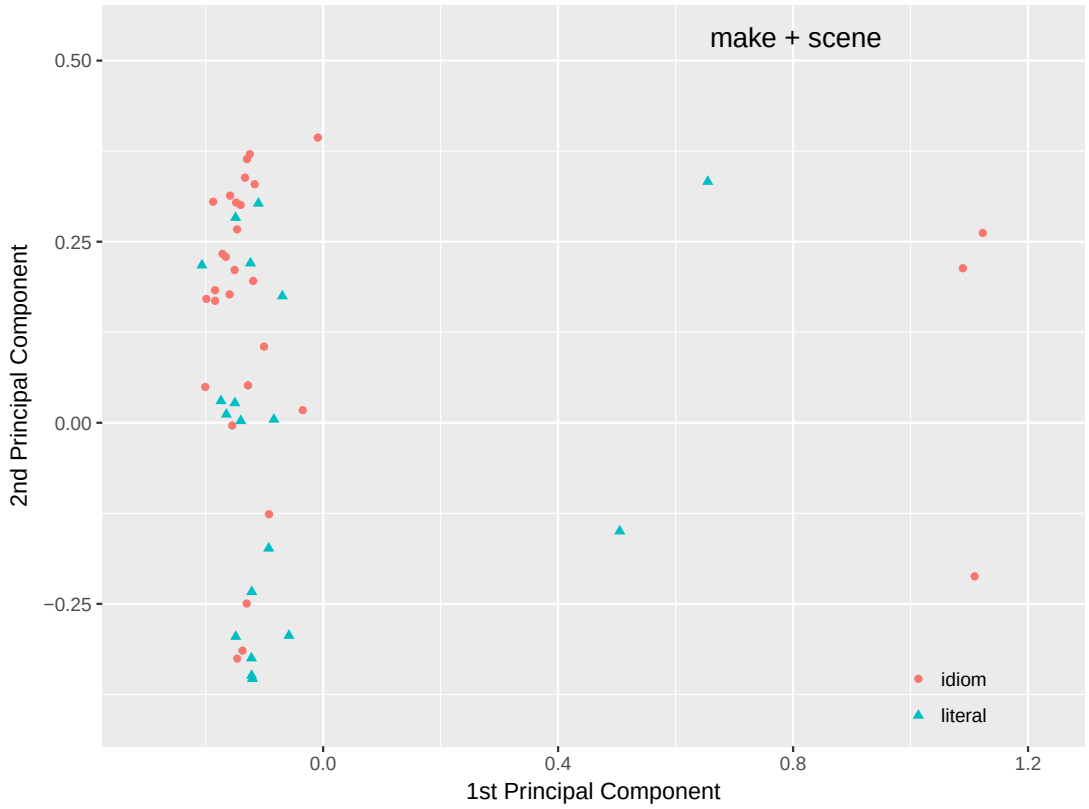

make + face
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

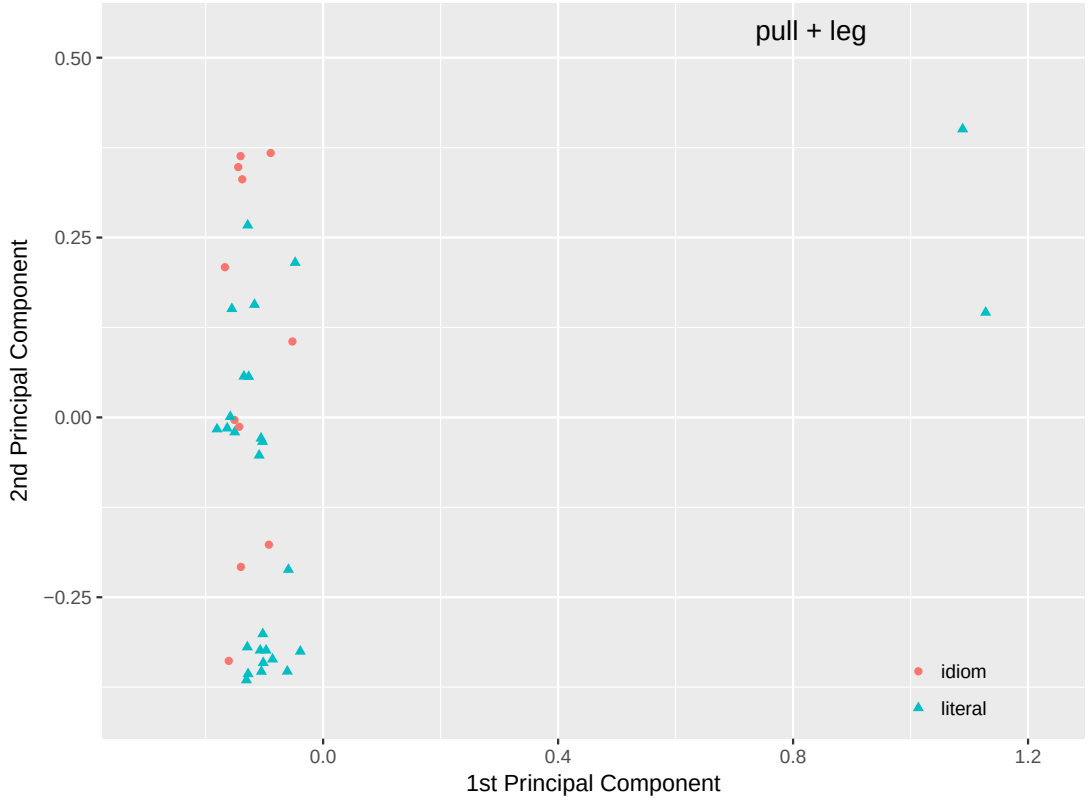

make + hay
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

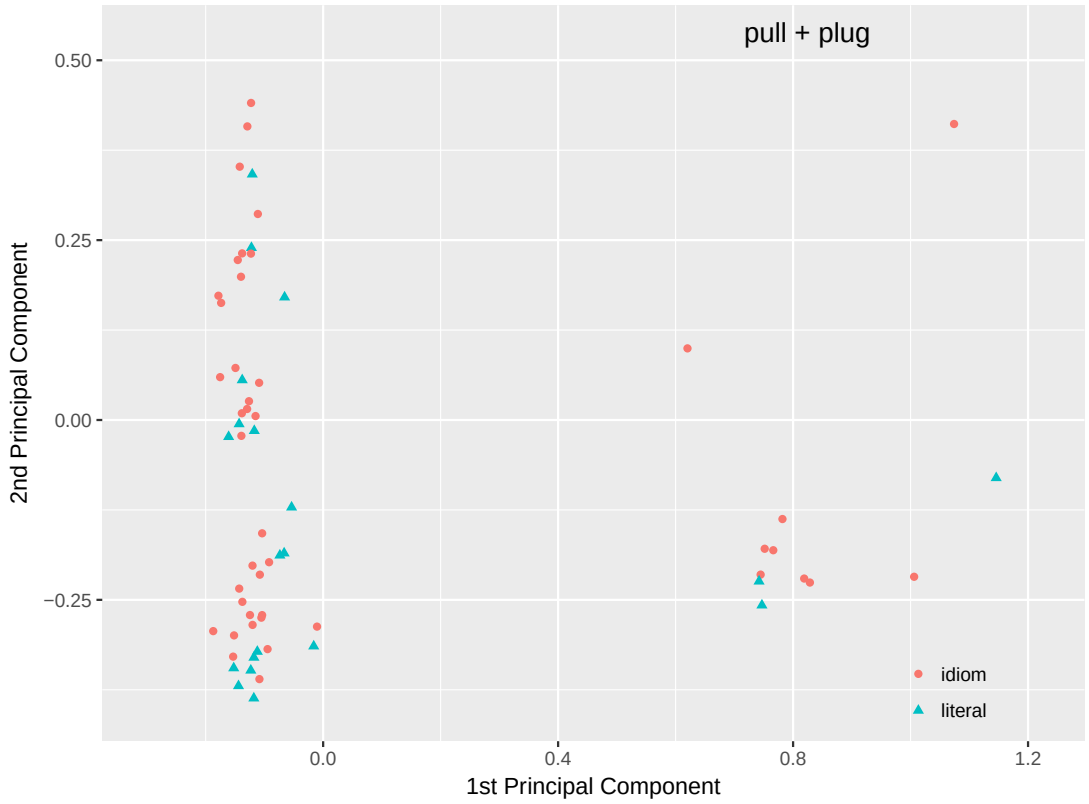

make + hit
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

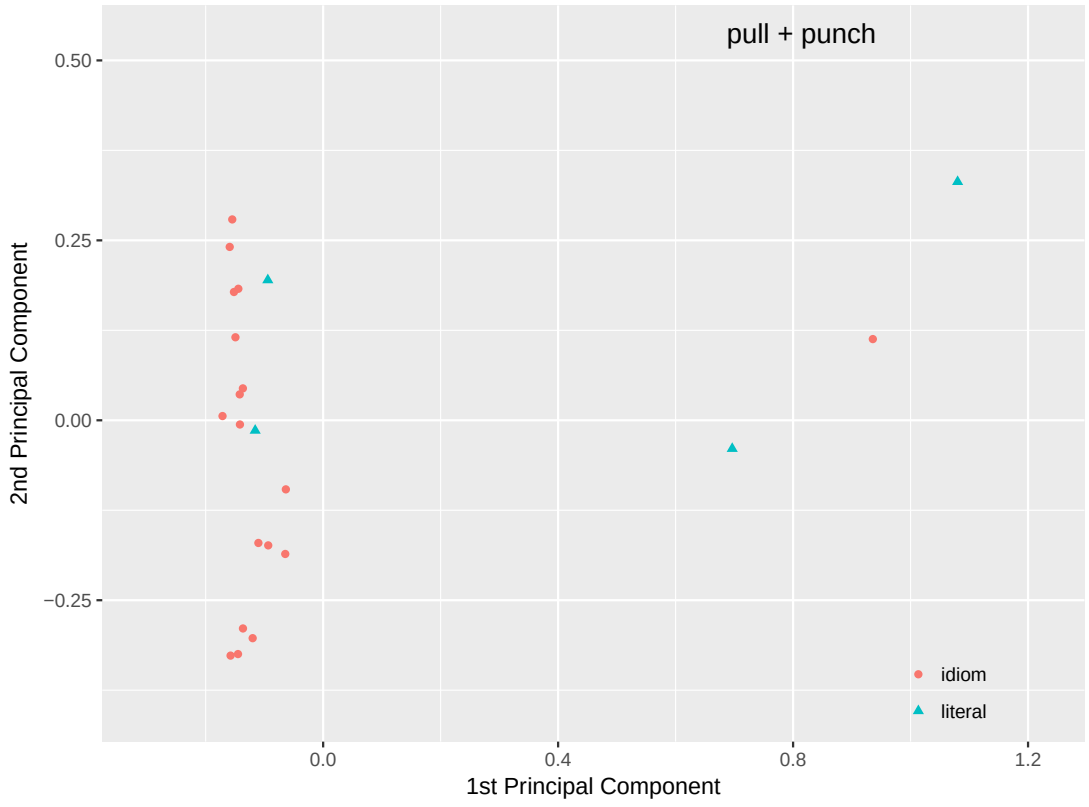

make + mark
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

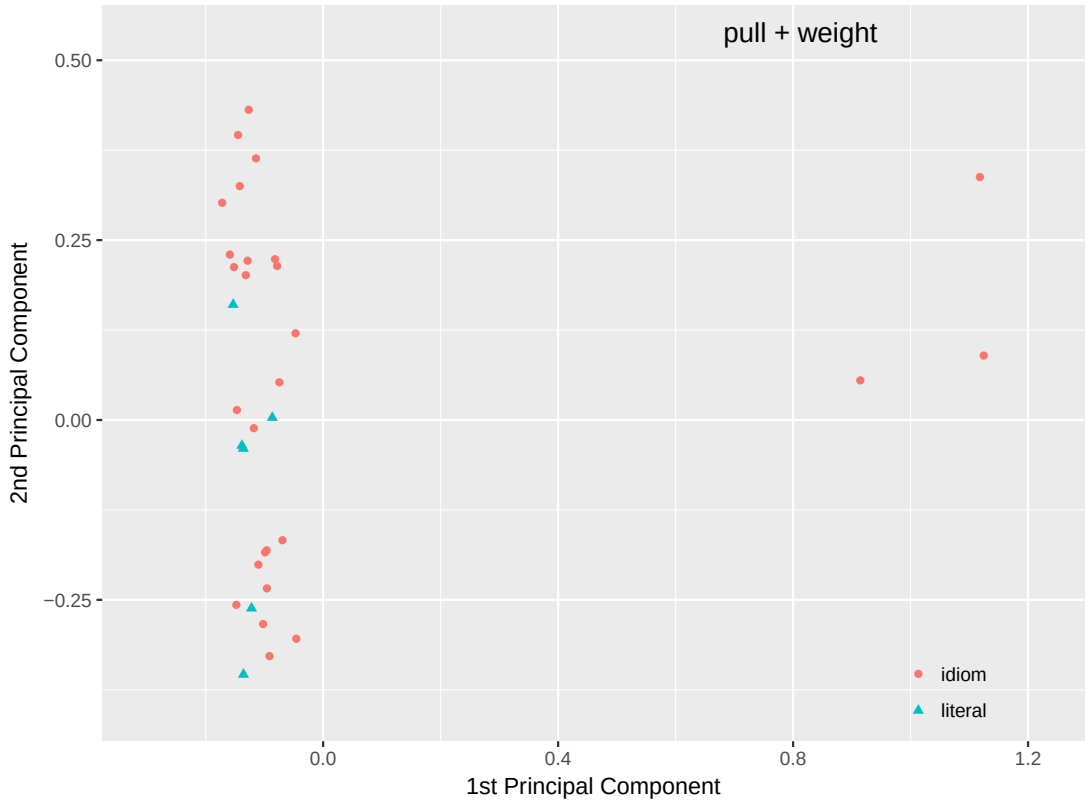

make + pile
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

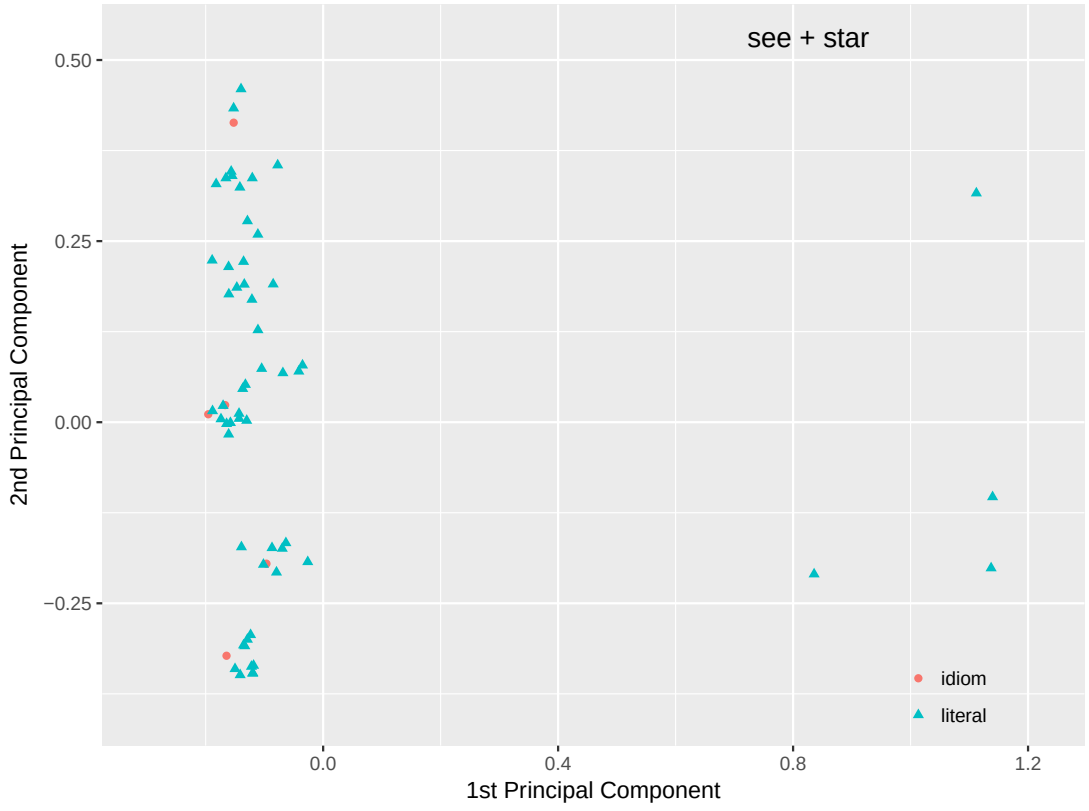

make + scene
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

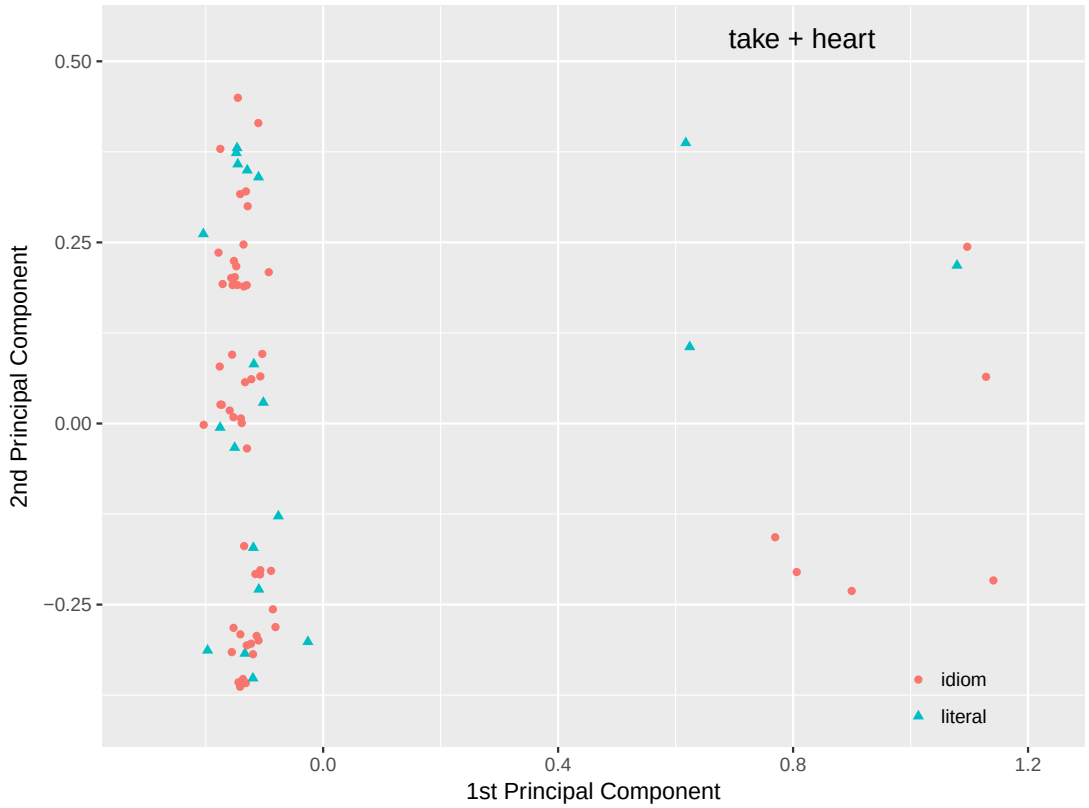

pull + leg
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal


pull + plug
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal


pull + punch
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal


pull + weight
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal


see + star
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal


take + heart
0.50
0.25
0.00
−0.25
2nd Principal Component
0.0
0.4
0.8
1.2
1st Principal Component
idiom
literal

Bibliography

Aziz, W. and Specia, L. (2011). Fully automatic compilation of a portuguese-english and portuguese-spanish parallel corpus for statistical machine translation. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology (STIL'2011)*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'2015)*.

Bai, M.-H., You, J.-M., Chen, K.-J., and Chang, J. S. (2009). Acquiring translation equivalences of multiword expressions by normalized correlation frequencies. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 478–486.

Baldwin, T. and Kim, S. N. (2010). Multiword expressions. In Indurkhya, N. and Damerau, F. J., editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL.

Bannard, C. (2007). A measure of syntactic flexibility for automatically identifying multiword expressions in corpora. In *Proceedings of the Workshop on a Broader Perspective on Multiword Expressions (MWE '07)*, pages 1–8.

Biber, D., Conrad, S., and Reppen, R. (1998). *Corpus linguistics: Investigating language structure and use*. Cambridge University Press.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post,

M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187.

Bouamor, D., Semmar, N., and Zweigenbaum, P. (2011). Improved Statistical Machine Translation Using MultiWord Expressions. In *Proceedings of the International Workshop on Using Linguistic Information for Hybrid Machine Translation)*, pages 15–20.

Bouamor, D., Semmar, N., and Zweigenbaum, P. (2012). Identifying bilingual multi-word expressions for statistical machine translation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 674–679.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Brychcín, T. and Konopík, M. (2014). Semantic spaces for improving language modeling. *Computer Speech and Language*, 28(1):192–209.

Bungum, L., Gambäck, B., Lynum, A., and Marsi, E. (2013). Improving word translation disambiguation by capturing multiword expressions with dictionaries. In *Proceedings of the 9th Workshop on Multiword Expressions*, pages 21–30.

Burnard, L. (2007). Reference guide for the british national corpus (xml edition). Technical report, http://www.natcorp.ox.ac.uk/.

Cacciari, C. and Tabossi, P. (1988). The comprehension of idioms. *Journal of Memory and Language*, 27(6):668 – 683.

Cap, F., Nirmal, M., Weller, M., and Schulte im Walde, S. (2015). How to account for idiomatic german support verb constructions in statistical machine translation. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 19–28, Denver, Colorado. Association for Computational Linguistics.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Church, K., Gale, W., Hanks, P., and Hindle, D. (1991). Using statistics in lexical analysis. In *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 115–164. Erlbaum.

Cilibrasi, R. L. and Vitányi, P. M. (2007). The google similarity distance. *IEEE Transactions On Knowledge And Data Engineering*, 19(3):370–383.

Cook, P., Fazly, A., and Stevenson, S. (2008). The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Copestake, A., Lambeau, F., Villavicencio, A., Bond, F., Baldwin, T., Sag, I. A., and Flickinger, D. (2002). Multiword expressions: linguistic precision and reusability. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1942–1947.

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. (2017). Frustratingly Short Attention Spans in Neural Language Modeling. *Proceedings of the 5th International Conference on Learning Representations (ICLR'2017)*.

Fazly, A., Cook, P., and Stevenson, S. (2009). Unsupervised type and token identification of idiomatic expressions. In *Computational Linguistics*, volume 35, pages 61–103. The MIT Press.

Fazly, A. and Stevenson, S. (2006). Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–344.

Feldman, A. and Peng, J. (2013). Automatic detection of idiomatic clauses. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'13, pages 435–446.

Freitag, M. and Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Fritzinger, F., Weller, M., and Heid, U. (2010). A survey of idiomatic preposition-noun-verb triples on token level. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press Professional, Inc., San Diego, CA, USA.

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Gal, Y. and Ghahramani, Z. (2015). A theoretically grounded application of dropout in recurrent neural networks. *arXiv*, abs/1512.05287.

Grave, E., Joulin, A., and Usunier, N. (2017). Improving neural language models with a continuous cache. *Proceedings of The 5th International Conference on Learning Representations (ICLR'2017)*.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

Hardmeier, C. and Volk, M. (2009). Using Linguistic Annotations in Statistical Machine Translation of Film Subtitles. In *The 17th Nordic Conference of Computational Linguistics (NODALIDA 2009)*, pages 57–64.

Hashimoto, C. and Kawahara, D. (2008). Construction of an idiom corpus and its application to idiom identification based on wsd incorporating idiom-specific features. In *Proceedings of the conference on empirical methods in natural language processing*, pages 992–1001. Association for Computational Linguistics.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Hearne, M. and Way, A. (2011). Statistical machine translation: A guide for linguists and translators. *Language and Linguistics Compass*, 5(5):205–226.

Ho, W. Y., Kng, C., Wang, S., and Bond, F. (2014). Identifying idioms in chinese translations. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 716–721, Reykjavik, Iceland. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1391.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. volume 9, pages 1735–1780, Cambridge, MA, USA. MIT Press.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015a). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. (2015b). Montreal neural machine translation systems for wmt'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu,

Y. (2016). Exploring the limits of language modeling. *arXiv*, abs/1602.02410.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kelleher, J. D., Namee, B. M., and D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies*. MIT Press.

Keller, F. and Lapata, M. (2003). Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv*, abs/1412.6980.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3276–3284.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP'2004)*, pages 388–395.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Koehn, P., Arun, A., and Hoang, H. (2008). Towards better machine translation quality for the German-English language pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 139–142.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C.,

Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 48–54.

Kordoni, V. and Simova, I. (2014). Multiword expressions in machine translation. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Li, L. and Sporleder, C. (2010a). Linguistic cues for distinguishing literal and non-literal usages. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 683–691.

Li, L. and Sporleder, C. (2010b). Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 297–300, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98)*, pages 768–774.

Lin, D. (1999). Automatic identification of non-compositional phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 317–324.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Luong, M.-T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, pages 114–119.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. *5th International Conference on Learning Representations (ICLR'2017)*.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016). Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas. Association for Computational Linguistics.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *The 2013 Conference of the North Americal Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751.

Muzny, G. and Zettlemoyer, L. S. (2013). Automatic idiom identification in wiktionary. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1417–1421.

Nerima, L., Foufi, V., and Wehrli, E. (2017). Parsing and mwe detection: Fips at the parseme shared task. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 54–59, Valencia, Spain. Association for Computational Linguistics.

Nunberg, G., Sag, I. A., and Wasow, T. (1994). Idioms. *Language*, 3(70):491–538.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA. Association for Computational Linguistics.

Okita, T. (2011). *Word Alignment and Smoothing Method in Statistical Machine Translation: Noise, Prior Knowledge and Overfitting*. PhD thesis, Dublinc City University.

Okuma, H., Yamamoto, H., and Sumita, E. (2008). Introducing Translation Dictionary Into Phrase-based SMT. In *IEICE - Transactions on Information and Systems*, volume E91-D, pages 2051–2057.

Oualil, Y., Singh, M., Greenberg, C., and Klakow, D. (2016). Long-short range context neural networks for language modeling. In *Proceedings*

of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1473–1481, Austin, Texas. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Peng, J., Feldman, A., and Vylomova, E. (2014). Classifying idiomatic and literal expressions using topic models and intensity of emotions. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2019–2027.

Petukhova, V., Agerri, R., Fishel, M., Georgakopoulou, Y., Penkale, S., del Pozo, A., Maucec, M. S., Volk, M., and Way, A. (2012). SUMAT: Data Collection and Parallel Corpus Compilation for Machine Translation of Subtitles. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 21–28.

Phillips, A. and Davis, M. (2009). Tags for Identifying Languages. RFC 5646.

Press, O. and Wolf, L. (2016). Using the output embedding to improve language models. *arXiv*, abs/1608.05859.

Ramisch, C., Villavicencio, A., and Boitet, C. (2010). Web-based and combined language models: A case study on noun compound identification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1041–1049, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ren, Z., Lu, Y., Cao, J., Liu, Q., and Huang, Y. (2009). Improving statistical machine translation using domain bilingual multiword expressions. In *Proceedings of the 2009 Workshop on Multiword Expressions, ACL-IJCNLP 2009*, pages 47–54.

Riehemann, S. (2001). *A Constructional Approach to Idioms and Word Formation*. PhD thesis, Stanford University.

Rommers, J., Dijkstra, T., and Bastiaansen, M. C. M. (2013). Context-dependent semantic processing in the human brain: Evidence from

idiom comprehension. *Journal of Cognitive Neuroscience*, 25(5):762–776.

Rondon, A., Caseli, H., and Ramisch, C. (2015). Never-ending multiword expressions learning. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 45–53, Denver, Colorado. Association for Computational Linguistics.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword Expressions: A Pain in the Neck for NLP. In *Computational Linguistics and Intelligent Text Processing: Third International Conference: CICLing-2002, Lecture Notes in Computer Science*, volume 2276, pages 1–15.

Salton, G. D., Ross, R. J., and Kelleher, J. D. (2014a). An Empirical Study of the Impact of Idioms on Phrase Based Statistical Machine Translation of English to Brazilian-Portuguese. In *Third Workshop on Hybrid Approaches to Translation (HyTra) at 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 36–41.

Salton, G. D., Ross, R. J., and Kelleher, J. D. (2014b). Evaluation of a substitution method for idiom transformation in statistical machine translation. In *The 10th Workshop on Multiword Expressions (MWE 2014) at 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 38–42.

Salton, G. D., Ross, R. J., and Kelleher, J. D. (2016). Idiom token classification using sentential distributed semantics. In *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics*, pages 194–204.

Salton, G. D., Ross, R. J., and Kelleher, J. D. (2017a). Attentive language models. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Salton, G. D., Ross, R. J., and Kelleher, J. D. (2017b). Idiom type identification with smoothed lexical features and a maximum margin classifier. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP'2017)*, pages 642–651.

Schwenk, H., Rousseau, A., and Attik, M. (2012). Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19.

Senaldi, M. S. G., Lebani, G. E., and Lenci, A. (2016). Lexical variability and compositionality: Investigating idiomaticity with distributional semantic models. In *Proceedings of the 12th Workshop on Multiword Expressions, MWE@ACL 2016, Berlin, Germany, August 11, 2016.*

Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh Neural Machine Translation Systems for WMT 16. In *Proc. of the First Conference on Machine Translation (WMT16)*, Berlin, Germany.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

Shutova, E., Sun, L., and Korhonen, A. (2010). Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1002–1010.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Spearman, C. (1907). Demonstration of formulæfor true measurement of correlation. *The American Journal of Psychology*, 18(2):161–169.

Sporleder, C. and Li, L. (2009). Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762.

Sporleder, C., Li, L., Gorinski, P., and Koch, X. (2010). Idioms in context: The idix corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC-2010)*, pages 639–646.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Tang, K. (2012). A 61 Million Word Corpus of Brazilian Portuguese Film Subtitles as a Resource for Linguistic Research. In *UCL Working Papers in Linguistics 24*, volume 24, pages 208–214.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Tran, K. M., Bisazza, A., and Monz, C. (2016). Recurrent memory network for language modeling. *arXiv*, abs/1601.01272.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, (37):141–188.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Vega-Moreno, R. E. (2001). Representing and processing idioms.

Vilar, D., Xu, J., D'haro, L., and Ney, H. (2006). Error analysis of statistical machine translation output. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy. European Language Resources Association (ELRA). ACL Anthology Identifier: L06-1244.

Villavicencio, A., Bond, F., Korhonen, A., and McCarthy, D. (2005). Editorial: Introduction to the special issue on multiword expressions: Having a crack at a hard nut. *Computer Speech and Language*, 19(4):365–377.

Villavicencio, A. and Copestake, A. (2004). The nature of idioms. In *LinGO Working Paper No. 2002-04*.

Villavicencio, A., Copestake, A., Waldron, B., and fabre Lambeau (2004). Lexical encoding of mwes. In *Proceedings of the Workshop on Multiword Expressions: Integrating Processing (MWE '04)*, pages 80–87.

Xiong, D., Zhang, M., and Li, H. (2010). Learning translation boundaries for phrase-based decoding. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 136–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057.

Zaki, M. J. and Meira Jr., W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv*, abs/1409.2329.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Zhang, H., Quirk, C., Moore, R. C., and Gildea, D. (2008). Bayesian learning of non-compositional phrases with synchronous parsing. In

Proceedings of ACL-08: HLT, pages 97–105, Columbus, Ohio. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv*.