Contextualised Approaches to Embedding Word Senses

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Master of Science (Research) in Computer Science

at

The University of Waikato

by

Alan Ansell

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2020

Abstract

Vector representations of text are an essential tool for modern Natural Language Processing (NLP), and there has been much work devoted to finding effective methods for obtaining such representations. Most previously proposed methods derive vector representations for individual words, known as word embeddings. While word embeddings have enabled considerable advances in NLP, they have a significant theoretical drawback: many words have several, often completely unrelated meanings; it seems dubious to conflate these multiple meanings into a single point in semantic space.

This drawback has inspired an alternative, "multi-sense" approach to representing words. In this approach, rather than learning a single vector for each word, multiple vectors, or "sense embeddings," are learned corresponding to the individual meanings of the word. While this approach has not in general surpassed the word embedding approach, it has proved beneficial for a number of tasks such as word similarity estimation and word sense induction.

One of the most significant recent advances in NLP has been the development of "contextualised" word embedding models. Whereas word embeddings model the semantic properties of words in isolation, contextualised models represent the meanings of words in context. This enables them to capture some of the vast array of linguistic phenomena that occur above the word level.

I propose a number of new methods for learning sense embeddings which exploit contextualised techniques, based on the underlying hypothesis that the probability of a word occurring in a given context is equal to the sum of the probabilities of its individual senses occurring in the context. I first validate this hypothesis by using it to derive a simple method for learning sense embeddings inspired by the Skip-gram model. I then present a method for extracting sense embeddings from a contextualised word embedding model. Finally I propose an end-to-end model for learning sense embeddings, and show that it comprehensively outperforms previous sense embedding models on the task of word sense induction, a standard task for evaluation of such models. To demonstrate the model's flexibility I apply it to some other word-sense related tasks with good results.

Acknowledgements

I would like to express my heartfelt appreciation for the support and guidance of my supervisors Prof. Bernhard Pfahringer and Dr. Felipe Bravo-Marquez. Working with you has been academically rewarding and lots of fun too!

I would also like to thank my family and Shibani; your support and encouragement have allowed me to achieve much more than I could have alone.

Contents

Chapter 1

Introduction

1.1 Overview

The purpose of this research is to advance the study of vector representations for word senses, known as *sense embeddings*, by developing new, unsupervised methods for learning sense embeddings which exploit recent contextualisation techniques. The key assumption which enables these techniques to be applied is that the probability of a word occurring in a given context is equal to the sum of the probabilities of its individual senses occurring in the context. I show that the proposed methods outperform previous sense embedding methods by a large margin on word sense induction, a standard evaluation task, and can be easily applied to other word-sense related tasks for which unsupervised sense embeddings have not been used previously.

1.2 Background

1.2.1 Word Embeddings

Mathematical representations of linguistic elements such as words and sentences have been at the heart of many techniques in Natural Language Processing (NLP) since the inception of the field. Vectors of real numbers have been an especially important form of representation in NLP.

The earliest use of vector representations in NLP was to provide a summary of document content for the purpose of information retrieval (Salton et al., 1975). However the majority of work on vector representations in NLP has been dedicated to *word vectors*, also known as *word embeddings*. An embedding is a mapping from a (potentially large) set of discrete objects to a corresponding set of low-dimensional real vector representations, in which those objects which are more similar tend to be closer together in the vector space. Figure 1.1 illustrates this property of word embeddings.

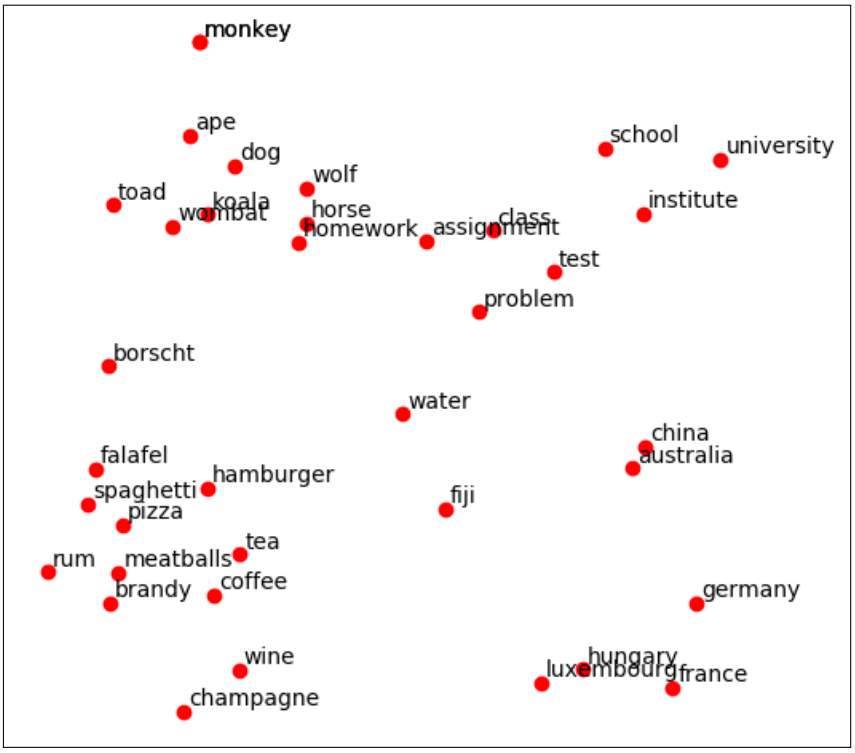


Figure 1.1: Visualisation of selected 100-dimensional GloVe (Pennington et al., 2014) word embeddings, reduced to two dimensions using principal component analysis. The diagram illustrates the property of word embeddings that words with similar meanings tend to appear near to each other in the vector space.

The *distributional hypothesis* (Harris, 1954; Firth, 1957) states that words which occur in the same contexts tend to have similar meanings, and can be seen as the basis of most word embedding methods. Of particular relevance is the class of prediction-based word embedding methods, which train their word embeddings by learning which words are likely to appear in a given context. The first such method was proposed by Bengio et al. (2000), where word embeddings were learned through

neural language modelling. The use of word embeddings in NLP became almost universal with the release of the prediction-based *word2vec* system (Mikolov et al., 2013a), and the slightly later GloVe method (Pennington et al., 2014). The areas in which these methods enabled advances included question answering (Seo et al., 2016), word sense disambiguation (Raganato et al., 2017b), coreference resolution (Lee et al., 2017) and dependency parsing (Weiss et al., 2015). A more detailed background on word embeddings will be given in Chapter 2.

1.2.2 Sense Embeddings

Despite the practical success of word embeddings, it has long been recognised that they have a significant theoretical drawback (Schütze, 1998): many words are *polysemous*, that is, they have several meanings, or *senses*. It therefore seems questionable to represent every word, regardless of how many meanings it has, as a single point in semantic space. This problem has been termed the "meaning conflation deficiency" (Camacho-Collados and Pilehvar, 2018), and is worsened in view of the Principle of Economical Versatility of Words (Zipf, 1950), which states that it is the most frequent words that tend to have more senses.

The meaning conflation deficiency has the practical consequence that polysemy causes distortion in word embeddings: for instance, we would find the unrelated words *left* and *wrong* unreasonably close in the vector space due to their similarity to two different senses of the word *right*, an effect noted by Neelakantan et al. (2014). Intuitively we would expect this problem to hamper the semantic understanding of word embedding models. Indeed, Yaghoobzadeh and Schütze (2016) suggest that word embedding models can have difficulties distinguishing which sense of an ambiguous word applies in a given context.

This drawback of word embeddings has inspired a number of models which attempt to obtain embeddings for individual word *senses*, referred to as *sense embeddings*. Sense embedding methods generally fall into one of two categories: *unsupervised* methods, in which sense embeddings are learned from unlabelled cor-

pora[1], and *knowledge-based* methods, which exploit lexical resources created by human experts, such as dictionaries. Unsupervised approaches have the advantage that such resources are unavailable for many languages.

In addition to the drawbacks associated with the meaning conflation deficiency, there are a number of applications motivating the interest in sense embeddings. Sense embeddings have been used to obtain good results on word sense induction (WSI) (Qiu et al., 2016; Song et al., 2016; Arora et al., 2018), and knowledge-based sense embeddings have been applied to word sense disambiguation (WSD) as well (Vial et al., 2017); these tasks are relevant to machine translation, information retrieval and information extraction (Agirre and Edmonds, 2006).

Another interesting use case is the automatic construction of lexical resources (Neale, 2018). I will also suggest a potential application to unsupervised machine translation.

More background on sense embeddings will be given in Chapter 3.

1.2.3 Contextualised Models

One of the most significant recent developments in NLP has been the emergence of contextualized models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). Contextualised models, or *contextualisers* (a term coined by Liu et al.), are instances of a successful application of *transfer learning* in NLP. Given a sequence of input words, such as a sentence, contextualised models output a corresponding sequence of *contextualised word embeddings*. While word embeddings represent the meaning of words in isolation, contextualised word embeddings represent word meanings in context by capturing some of the linguistic phenomena which operate above the word level.

Peters et al. showed that the contextualised representations produced by their ELMo contextualised model could be used to significantly improve the state-of-the-art on a wide range of tasks including question answering, coreference resolution, named entity resolution and sentiment analysis.

[1] A *corpus* is a large body of text used to train an NLP model.

Subsequently, contextualisers have been incorporated into many state-of-the-art systems for specific tasks, including WSD (Huang et al., 2019; Vial et al., 2019) and WSI (Amrami and Goldberg, 2018, 2019). Further background of contextualisation methods is given in Chapter 4.

1.3 Aim

The success of contextualised models has had a tremendous impact on the area of semantic representation, and the claim that they capture the meaning of words in context raises the question of whether there is still value in learning discrete sense representations.

However, contextualised models employ a single, fixed input embedding for each word, and are therefore still subject to the meaning conflation deficiency. Furthermore it could be argued that it is inefficient to have the same representation size for all words regardless of how diverse their range of senses is. Another drawback is that before they can be applied to word sense-related tasks, a time-consuming adaptation step such as fine-tuning (Huang et al., 2019) or clustering to learn discrete senses (Amrami and Goldberg, 2019) is generally required. For each new task, research is required to develop a specialised technique for doing so.

The aim of this research has been to develop better methods for unsupervised learning of sense representations. It seemed natural to focus on applying contextualisation techniques to the problem, as they are the most powerful recent techniques for semantic representation, and have not previously been used for learning sense embeddings.

1.4 Contributions

1.4.1 Word-Sense Probability Hypothesis

Contextualised models are typically trained on a language modelling task of some description, in which the objective is to maximise the model's estimated probability

of words that occurred in the corpus given their surrounding context.

The ideal task for training a contextualised sense model would be "sense modelling," where the model must predict both the word which appears in a context and its sense, but this is impossible because in an unsupervised setting, the only ground truth available is the words themselves, not their senses; and while some sense-annotated corpora are available for English, these are not large enough to train a strong contextualised model. To enable prediction-based training of contextualised sense models, we need something which links probabilities of words with probabilities of individual word senses.

The probability of a word w occurring in a given context c (a concept which will be discussed in more detail in Section 2.1) can be written as $\mathbb{P}(w \mid c)$. Here we use the term "context" generally. For instance, in the sentence *"I am cooking dinner."*, the context of the word "cooking" could refer to just the left side context *"I am..."*, the left and right side context *"I am... dinner."*, or even the bag-of-words representation of the context {*I*, *am*, *dinner*, .} in which word order is ignored.

Suppose that each word w in the vocabulary V has a set of meanings S_w which it can take on according to the context. Then the probability of w occurring in a context c is equal to the sum of the probabilities of each of its individual meanings occurring in c:

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \mathbb{P}(s \mid c), \tag{1.1}$$

where $\mathbb{P}(s \mid c)$ is the probability of w occurring in context c with meaning s. Assuming that S_w contains all meanings which w can have, and w has only one meaning in any context, Equation 1.1 follows from the partition theorem of probability theory.

For example, intuitively we might have $S_{bank} = \{$bank: financial institution, bank: side of a river$\}$. In the context *"I went to the _ ."*, the probability that the missing word is *bank* is equal to the probability that the missing word is *bank* with the financial sense plus the probability that it is *bank* with the river sense.

Equation 1.1 provides the necessary link between word and meaning probabilities, and underlies all of the methods described in this thesis.

1.4.2 Methods

In Chapter 5, I describe a sense embedding method based on the Skip-gram model of Mikolov et al. (2013a). This method was developed early on in my research and served as a "proof-of-concept" of the hypothesis in 1.4.1. I provide a qualitative analysis showing that the method does indeed learn reasonable sense embeddings.

While contextualised word embedding methods have proven effective at representing the meanings of words in context, they do not model polysemy explicitly. In Chapter 6, I present Maximum Likelihood Contextual Clustering (MLCC), a method for obtaining explicit sense embeddings from a contextualised word embedding model. I show that MLCC can be used to obtain much better results than previous sense embedding methods on word sense induction (WSI), a standard task for evaluating such models. I also demonstrate that it even outperforms the state-of-the-art specialised WSI method when comparably sized models are used.

In Chapter 7 I propose PolyLM, which in contrast to the contextual classification method is an end-to-end model which learns sense embeddings for all words simultaneously. By using only sense embeddings, PolyLM aims to avoid meaning conflation as much as possible. I show that PolyLM is similarly effective to contextual classification at WSI, and can also be simply and effectively applied to a number of other word sense-related tasks to which sense embeddings have not been applied before. PolyLM can be used with any contextualiser and therefore the model will be able to be used to obtain better sense embeddings as contextualisation techniques improve.

Chapter 2

Prediction-based Word Embedding Models

Prediction-based word[1] embedding models are those which, during training, use word embeddings to make local predictions within the corpus, such as predicting occurrences of words given their contexts, or predicting the context given a word contained within it. Count-based models on the other hand utilise global co-occurrence statistics (Almeida and Xexéo, 2019). For instance, the popular GloVe method (Pennington et al., 2014) operates on a matrix where the i, jth entry denotes the number of times words i and j occur within a certain distance of each other in the corpus.

Since all of the methods presented in this thesis are prediction-based, this chapter focuses on the background of prediction-based methods. In particular, we will discuss neural language modelling and the Skip-gram model, which will be built upon in later chapters.

[1] In the context of word embeddings, and NLP in general, the word *word* is usually used as a synonym for *token*, a single, indivisible element of text, such as a word (in the everyday sense), number or punctuation mark. Word embedding models typically treat all types of tokens equivalently.

2.1 Probability and Language

A fundamental concept in prediction-based word embedding models is that of word probability. Word probability is often thought of in relation to a corpus. The simplest notion of word probability is *unigram* probability. The unigram probability p_w of a word w is the probability that a word randomly sampled from the corpus is w. Formally,

$$p_w = \frac{f_w}{\sum_{v \in V} f_v}, \tag{2.1}$$

where V is the vocabulary[2] and f_v denotes the number of times token v occurs in the corpus.

The concept of word probability is more interesting when it is conditioned on a context. Consider the following question for example: if a sentence starts "Why did the...," then how likely is it that the next word will be "chicken"? This probability can be written informally as

$$\mathbb{P}(\text{“chicken”} \mid \text{“Why did the...”}). \tag{2.2}$$

In general, the probability of a word w occurring in a context c can be written as

$$\mathbb{P}(w \mid c). \tag{2.3}$$

The use of probabilities in linguistics has been dismissed by the pioneering linguist Chomsky (1969), writing *"But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term."*. Chomsky (1957) provides evidence of this in the form of the following two sentences:

1. *Colorless green ideas sleep furiously.*

[2]The *vocabulary* is the set of "recognised" tokens. To keep its size manageable, it is typically chosen to be a proper subset of the full set of tokens which occur in the corpus, often the k most frequent words to ensure the largest possible coverage of the corpus. Words not in the vocabulary are often replaced with a special "unknown" token which is also added to the vocabulary. The symbol V always refers to the vocabulary in this thesis.

2. *Furiously sleep ideas green colorless.*

While both sentences are nonsensical, the first is a grammatical sentence of English and the second is not. Since it is almost certain that neither has occurred before in human discourse, Chomsky argues that a statistical model ought to assign both a probability of zero, failing to make any distinction between the sentences despite this fundamental difference.

Chomsky's argument is predicated on a frequentist view of probability, where the probability of a sequence of words is proportional to the number of times that sequence has occurred previously. However, the statistical models used in NLP are capable of generalising to unseen sequences of words. As we will see, the use of word embeddings is one way to enable effective generalisation of this type.

2.2 Neural Language Modelling

A language model is a statistical model which, given the beginning of a sequence of words (e.g. *"Why did the"*, if the full sequence is *"Why did the chicken cross the road?"*), outputs a probability distribution over the vocabulary corresponding to its estimate of the probability of each word in the vocabulary being the next to occur in the sequence. Formally, given words $w_1, w_2, ..., w_{t-1} \in V$, a language model outputs a probability distribution over V for w_t. Language modelling can be viewed as a multiclass classification problem, where the examples are sequences of words, and the classes are the words in the vocabulary. However language modelling is considered an unsupervised task because examples can be constructed from an unlabelled text corpus.

For a long time the dominant approach to language modelling was the *N-gram* approach, introduced by Shannon (1951). An N-gram is a unique sequence of N words. The probability predicted by an N-gram model of a given word occurring next in a sequence is the probability observed in the corpus of the word occurring, conditioned on the previous $N-1$ tokens. A serious drawback of N-gram models is that they do not generalise well to previously unseen sequences; they tend to assign

high probability only to sequences that have occurred in the training corpus, but as the number of plausible N-grams is exponentially large, even a large corpus can only contain a small subset of them.

A competing approach, neural language modelling, was proposed in 2000 by Bengio et al.. The word *neural* denotes the use of continuous representations of word sequences rather than discrete representations such as N-grams, and specifically the use of artificial neural networks.

Bengio et al.'s model consists of two components: an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, where d is the embedding dimensionality, and a probability function $\boldsymbol{f}$, a neural network which takes as input the embeddings of the previous words in the sequence and outputs a estimated probability distribution over next words.

In order to obtain a fixed-sized input to $\boldsymbol{f}$, Bengio et al.'s model considers only the n previous words $w_{t-n}, w_{t-n+1}, ..., w_{t-1}$ when predicting w_t. The input representation $\boldsymbol{x} \in \mathbb{R}^{nd}$ is found by concatenating the embeddings of these words, i.e. $\boldsymbol{x} = [\boldsymbol{e}_{w_{t-n}}; \boldsymbol{e}_{w_{t-n+1}}; ...; \boldsymbol{e}_{w_{t-1}}]$, where $\boldsymbol{e}_i$ is the ith row of E. $\boldsymbol{f}$ is a feed-forward neural network with a single hidden layer (with skip connection) and a softmax output layer yielding a vector of length $|V|$, the elements of which correspond to the words in the vocabulary:

$$\boldsymbol{y} = \boldsymbol{b} + W\boldsymbol{x} + U \tanh(\boldsymbol{d} + H\boldsymbol{x}) \tag{2.4}$$

$$\boldsymbol{f} = \text{softmax}(\boldsymbol{y}). \tag{2.5}$$

The parameters $\Theta = (\boldsymbol{b}, \boldsymbol{d}, W, U, H, E)$ are adjusted through stochastic gradient ascent to maximise the mean log-likelihood J of the words contained in the corpus, i.e.

$$J(\Theta) = \frac{1}{T} \sum_t \log[\boldsymbol{f}(w_{t-n}, ..., w_{t-1}\,; \Theta)]_{w_t} + R(\Theta), \tag{2.6}$$

where T is the total number of words in the corpus and $R(\Theta)$ is a regularization term.

Bengio et al.'s model achieved state-of-the-art performance by a significant margin over the previous N-gram models. The use of word embeddings enabled the model to generalise in ways the N-gram models could not. Bengio et al. offer the following explanation for this:

> If we knew that *dog* and *cat* played similar roles (semantically and syntactically), and similarly for (*the,a*), (*bedroom,room*), (*is,was*), (*running,walking*), we could naturally generalise (i.e. transfer probability mass) from
>
> *The cat is walking in the bedroom*
> to *A dog was running in a room*
> and likewise to *The cat is running in a room*
> *A dog is walking in a bedroom*
> *The dog was walking in the room*
> ...
>
> and many other combinations. In the proposed model, it will so generalise because "similar" words are expected to have a similar feature vector, and because the probability function is a smooth function of these feature values, a small change in the features will induce a small change in the probability. Therefore, the presence of only one of the above sentences in the training data will increase the probability, not only of that sentence, but also of its combinatorial number of "neighbors" in sentence space (as represented by sequences of feature vectors).

Note that the term *feature vector* is synonymous with *embedding*.

2.3 Skip-gram Model

2.3.1 Motivation

While Bengio et al.'s work demonstrated the potential of word embeddings, it used them only as a means for obtaining the best possible language modelling performance. Subsequent work introduced the idea that word embeddings trained on one task could be applied to another. Collobert and Weston (2008) proposed an architecture for jointly training word embeddings on several tasks simultaneously. They showed that when their model was trained on both language modelling and semantic role labelling (Palmer et al., 2005), or SRL, its performance on SRL was significantly better than when it was only trained on SRL. This illustrated that word embeddings can be used to capture features of words that are transferable across applications.

A breakthrough in transferable word embeddings came in the form of the Skip-gram model (Mikolov et al., 2013a,c), which was implemented in the *word2vec* software package[3]. The Skip-gram model prioritises simplicity and efficiency to enable rapid training on large corpora.

A significant drawback of language modelling methods is that for each word in the corpus, the model must predict a probability distribution over the entire vocabulary, which often has a size of the order of 100,000. This makes neural language models expensive to train - Bengio et al. trained their model for approximately three weeks using 40 CPUs. The Skip-gram model avoids this cost by framing the task of learning word embeddings as a binary classification problem. The Skip-gram model learns word embeddings which, given the "center" word of a context, allow it to best discriminate between those words which do and do not occur in the context.

2.3.2 Formulation

Unlike Bengio et al.'s model, the Skip-gram model learns *two* embeddings per word. The *input* vector $\boldsymbol{u}_i \in \mathbb{R}^d$ for word $i \in V$, represents word i when it is the *center* word. Likewise, *output* vector $\boldsymbol{v}_i \in \mathbb{R}^d$ represents word i when it is a context word. There are no hidden layers: a similarity score between two words i and j, denoting how likely j is to appear in the context of word i, is given by the dot product of their vectors, $\boldsymbol{u}_i^\top \boldsymbol{v}_j$.

The Skip-gram objective utilises *negative sampling*, based on *noise contrastive estimation* (NCE) (Gutmann and Hyvärinen, 2012). For each center word, the objective is to jointly maximise the estimated probability of the context words occurring, while minimising the probability of a number of randomly sampled "noise" words which did not occur in the context of the center word. The Skip-gram objective can be expressed as

$$J(c, o\ \boldsymbol{u}_:, \boldsymbol{v}_:) = \log \sigma(\boldsymbol{u}_c^\top \boldsymbol{v}_o) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n} \Big[\log \big(1 - \sigma(\boldsymbol{u}_c^\top \boldsymbol{v}_{w_i})\big) \Big], \tag{2.7}$$

[3] https://code.google.com/archive/p/word2vec/

where c is the center word, o is a word that actually occurred in the context of c, σ is the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

which maps real numbers to numbers in the interval $[0, 1]$ (which can be thought of as probabilities), k is the number of negative samples per true context word and $w_1, ..., w_k$ are randomly sampled words from distribution P_n that did not occur in the context of c. An obvious choice for the distribution P_n, which determines how often each word in the vocabulary is negatively sampled, would be to set each word w's negative sampling probability $P_n(w)$ to be its unigram probability p_w. However, Mikolov et al. (2013c) found that better word embeddings were obtained when rare words were sampled more frequently than they occur in the corpus as a whole. This was achieved by "smoothing" the unigram distribution by raising it to the power of $3/4$:

$$P_n(w) = \frac{p_w^{3/4}}{\sum_{v \in V} p_v^{3/4}}. \tag{2.9}$$

Unlike traditional language models, the Skip-gram model uses both the left and right side context around the center word. Those words within a window of width $m = 5$ in either direction around the center word were considered to be part of its context.

2.3.3 Properties

Mikolov et al. (2013d) proposed a task for evaluating the extent to which a set of word embeddings captures relevant linguistic features. The task is based on word *analogies*, such as

Man is to *woman* as *King* is to *Queen*.

Man/woman and *King/Queen* both differ only in the gender of the entities to which they refer. If there exists a direction in the word embedding space which corresponds to gender, we ought to be able to recover the "gender offset" by subtracting

the vector for *man* from the vector for *woman*. We would then hope that by adding the resulting vector to the vector for *King*, we would obtain a vector very similar to the vector for *Queen*.

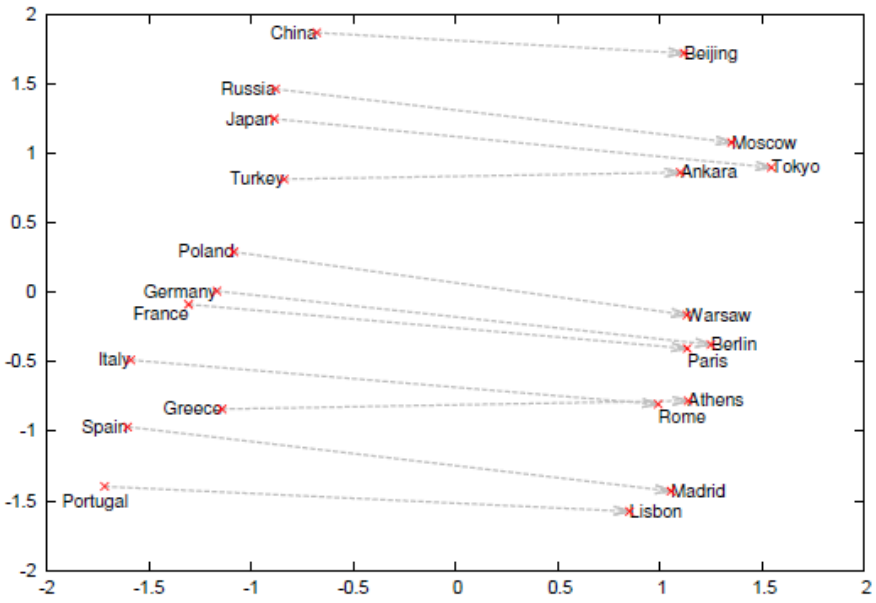


Figure 2.1: 1000-dimensional word2vec vectors for countries and their capital cities visualised using PCA. The diagram illustrates that there is a fairly regular vector offset between a country and its capital city; this effect enables word2vec vectors to accurately answer word analogy problems. Reproduced from Mikolov et al. (2013c).

Mikolov et al. (2013d,a) produced a dataset containing a number of analogies involving semantic relations such as gender and syntactic relations like tense (e.g. *walking* is to *walked* as *swimming* is to *swam*). Given three of the words in each analogy, the fourth is predicted as follows: taking the earlier analogy as an example, we calculate

$$\boldsymbol{u}_{\text{target}} = \boldsymbol{u}_{\text{woman}} - \boldsymbol{u}_{\text{man}} + \boldsymbol{u}_{\text{king}}. \tag{2.10}$$

We then find the word a whose vector is most similar to $\boldsymbol{u}_{\text{target}}$,

$$a = \underset{w \in V}{\operatorname{argmax}} \{ s(\boldsymbol{u}_w, \boldsymbol{u}_{\text{target}}),\ w \neq \text{man, woman, king} \}, \tag{2.11}$$

where s denotes cosine similarity:

$$s(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}. \tag{2.12}$$

If we find $a =$ queen, then the word embedding system is considered to have answered the instance correctly. Mikolov et al. (2013c) showed that their Skip-gram

model hugely outperformed previous word embedding models such as that of Collobert and Weston (2008) on this task, despite a much shorter training time.

2.4 Applications

It was shown prior to the development of the Skip-gram model that word embeddings trained on unlabelled data could be used to improve the performance of NLP systems on a range of tasks (Collobert and Weston, 2008; Turian et al., 2010). However, the availability of the high-quality pretrained embeddings produced by the Skip-gram model (Mikolov et al., 2013c) and the subsequent GloVe model (Pennington et al., 2014) enabled word embeddings to be fruitfully applied almost universally, including in parsing (Chen and Manning, 2014), question answering (Seo et al., 2016), sentiment analysis (Wang et al., 2016), and many other areas.

Chapter 3

Sense Embeddings

3.1 Motivation

3.1.1 Meaning Conflation Deficiency

A fundamental objection to the notion of word embeddings is that many words have more than one meaning, or *sense*. *Homonymous* words, such as *bark*, whose meanings are semantically unrelated, can be considered to consist of several, separate entries in the lexicon which happen to share the same spelling and phonological realisation. Therefore from a theoretical point of view it seems no more reasonable to represent with a single mathematical representation the meanings of the word *bark* (the sound made by a dog, and the outer layer of a tree) than it would to be to represent the words *meow* and *leaf* similarly, and yet this is what word embeddings do. This drawback is referred to as the *meaning conflation deficiency* (Camacho-Collados and Pilehvar, 2018). A similar but less severe effect results from the existence of *polysemous* words, or those which have several meanings which share some semantic similarity. The polysemous word *cell*, for instance, can refer to a small room such as a prison cell, or a structural unit of a living organism. The two meanings both refer to enclosed, self-contained units that are part of a larger structure. We will take the term "polysemy" to encompass both homonymy and polysemy in its strict sense.

The meaning conflation deficiency results in distortion in word embeddings. It

is common to estimate the degree of similarity between two words by comparing their vectors; recall for example the use of cosine distance for the word analogy task in Section 2.3.3. Suppose that d is a distance function, and that d(left, right) denotes the distance between the embeddings of the words *left* and *right*. The word *right* is closely related to both the words *left* and *wrong*, so we would expect both d(right, left) and d(right, wrong) to be small. But by the subadditive property of distance, we have

$$d(\text{left}, \text{wrong}) \leq d(\text{right}, \text{left}) + d(\text{right}, \text{wrong}). \tag{3.1}$$

Therefore we would expect a word embedding system to predict a relatively high degree of similarity between the words *left* and *wrong*, although they are in fact unrelated. This effect is illustrated in Figure 3.1.

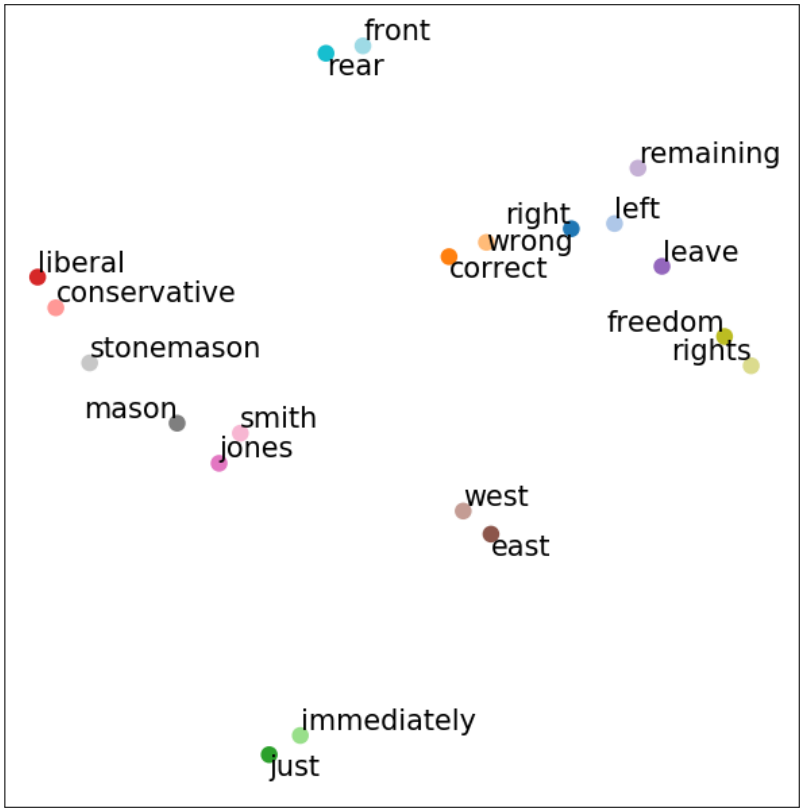


Figure 3.1: Word embeddings learned by MiniBERT (see Section 6.4.1) for a number of polysemous words, visualised using t-SNE (Maaten and Hinton, 2008) and adjustText (Flyamer, 2017). The occurrence of closely related polysemous words nearby in the word embedding space (i.e. *left* and *right*) causes unrelated words to be closer together (e.g. *left* and *wrong*) and related words to be further apart (e.g. *right* and *east*) than they otherwise would be.

Unfortunately there has been little work examining the extent to which the

meaning conflation deficiency harms the performance of NLP systems which utilise word embeddings. Li and Jurafsky (2015) proposed a system for learning embeddings for individual word senses ("sense embeddings") and a pipeline for substituting these into NLP systems in place of word embeddings. They found that for some natural language understanding tasks, their sense embeddings outperformed word embeddings of equal dimensionality, but that similar performance gains could be achieved more easily by increasing the word embedding dimensionality. Like Li and Jurafsky, I will provide comparisons against word embedding baselines where possible.

3.1.2 Applications

3.1.2.1 Word Sense Disambiguation and Induction

Regardless of the extent to which the meaning conflation deficiency hampers the semantic understanding of word embedding models in practice, there is good reason to be interested in deriving vector representations for word senses. Sense embeddings have been applied fruitfully applied to word sense disambiguation (WSD) (Vial et al., 2017) and word sense induction (WSI) (Qiu et al., 2016; Song et al., 2016; Bartunov et al., 2016; Arora et al., 2018). These tasks are relevant to machine translation, information retrieval and information extraction (Agirre and Edmonds, 2006), and will be discussed further in Section 3.2.

3.1.2.2 Construction of Lexical Resources

Another interesting use case is the automatic construction of lexical resources (Neale, 2018). While there are existing human-curated word sense inventories for English such as WordNet (Miller, 1995), these are expensive to create and are unavailable for most languages. Panchenko (2016) showed that sense embeddings learned using the model of Bartunov et al. (2016) could be linked with word senses contained in BabelNet (Navigli and Ponzetto, 2012) with a reasonable degree of precision, although the mapping struggled with recall. The methods presented in this thesis

achieve a significant improvement over Bartunov et al.'s in terms of WSI performance, so it seems reasonable to imagine that this approach to lexical resource construction might now be more feasible.

3.1.2.3 Multilingual NLP

The area of multilingual NLP is currently undergoing rapid development (Ruder et al., 2019). Of particular interest has been the success of systems for unsupervised machine translation (Artetxe et al., 2018, 2019), a task which requires the ability to translate between two languages without any parallel bilingual data. Underlying this advance is the use of cross-lingual word embeddings, where embeddings for words in two or more languages are aligned in the same semantic space. The dominant approach to learning cross-lingual word embeddings is to first learn monolingual embeddings for the two target languages, and then learn a function, often a linear projection (Mikolov et al., 2013b), which maps from one embedding space to the other. The objective when learning this function is to map the embeddings for a number of known synonyms in the two languages onto each other as accurately as possible. In a fully unsupervised context, these synonyms (known as the *seed vocabulary*) may for instance be numerals (1, 2, 3...), which can be relied upon to have the same meaning cross-lingually.

It would be entirely feasible to construct cross-lingual sense embeddings in the same manner, and I believe that this may be beneficial. As we have seen, polysemy distorts monolingual word embedding spaces. Furthermore, polysemy is certain to distort word embeddings spaces in different languages differently. Consider the Spanish word *sierra*, which can refer to both a mountain range or a saw (as in "a tool for cutting"). However, *saw* is also the past tense of the English word *see*. A cross-lingual word embedding projection which successfully linked the English word *saw* and the Spanish word *sierra* would also induce a spurious relationship between the concepts of vision and mountain ranges. Therefore it appears that the meaning conflation deficiency is doubly bad in a cross-lingual setting. This hypothesis is so far untested, but I think it warrants investigation.

3.1.2.4 Why Unsupervised Sense Embeddings?

Knowledge-based approaches to learning sense embeddings (Iacobacci et al., 2015; Pilehvar and Collier, 2016) exploit lexical resources such as BabelNet and WordNet; there are several advantages to unsupervised methods, that is, those that require only unlabelled corpora. Lexical resources are unavailable for most of the world's languages, and even in those languages for which they are available, the amount of unlabelled text available is much larger. Furthermore, such resources tend to be incomplete in ways that may affect real-world performance. For instance, WordNet has no entry for the surname meaning of the word "Cook," which is easily detected by unsupervised methods (although it does have an entry specifically for the explorer James Cook).

The applications mentioned in Sections 3.1.2.2 and 3.1.2.3 are particularly relevant for languages which lack lexical resources, the use of sense embeddings for the construction of lexical resources for obvious reasons, and their use for unsupervised machine translation for the reason that unsupervised machine translation is generally more useful for low-resource languages than for high-resource languages where supervised machine translation is possible.

3.2 Evaluation

Word and sense embeddings are difficult to interpret, so it is useful to have an array of techniques and tasks for evaluating them.

3.2.1 Nearest Neighbours

A simple method for visualising the content of an embedding is to list the words corresponding to the other embeddings to which it is closest ("neighbours") according to some similarity measure, typically cosine similarity. Table 3.1 shows some nearest neighbours reported in previous work on sense embeddings.

System	Word	Nearest Neighbours
Huang et al. (2012)	$bank_1$	corporation, insurance, company
	$bank_2$	shore, coast, direction
	$star_1$	movie, film, radio
	$star_2$	galaxy, planet, moon
	$cell_1$	telephone, smart, phone
	$cell_2$	pathology, molecular, physiology
	$left_1$	close, leave, live
	$left_2$	top, round, right
NP-MSSG (Neelakantan et al., 2014)	$apple_1$	pear, honey, pumpkin
	$apple_2$	microsoft, ibm, wordperfect
	fox_1	rabbit, squirrel, wolf
	fox_2	cbs, abc, nbc
	net_1	negative, total, transfer
	net_2	pre-tax, taxable, per
	net_3	ball, yard, fouled
	net_4	wnet, tvontorio, cable
	$rock_1$	granite, basalt, boulders
	$rock_2$	alternative, indie, pop/rock
Qiu et al. (2016)	$bank_1$	banking, lender, loan
	$bank_2$	river, canal, basin
	$bank_3$	slope, tilted, slant
	$apple_1$	macintosh, imac, blackberry
	$apple_2$	peach, cherry, pie
	$date_1$	birthday, birth, day
	$date_2$	appointment, meet, dinner
	fox_1	cbs, abc, nbc
	fox_2	wolf, deer, rabbit

Table 3.1: Nearest neighbours of sense embeddings learned by various methods.

3.2.2 Word Similarity

One quantitative technique for evaluating the quality of a set of word or sense embeddings is to compare the degree of similarity between embeddings for pairs of words with human judgements of their similarity. The automatic evaluation of semantic similarity is an important task in its own right, with applications to information retrieval (Varelas et al., 2005) and biomedical research (Pesquita et al., 2009).

The most popular word similarity dataset for evaluating sense embeddings is the Stanford Contextual Word Similarities (SCWS) dataset (Huang et al., 2012), which elicits particular word senses by providing a context for each word occurrence. The dataset consists of 2,003 pairs of sentences, with one word in each sentence specified as the "focus" word. Each pair is associated with 10 human judgements of the

semantic similarity of the two focus words in their context sentences, on a scale of 0 to 10. Performance is measured using the Spearman's ρ correlation between the similarity predicted by the embedding system and the average human rating.

While the use of word similarity as a task for evaluating sense embeddings has been common in the past, I have avoided it in this research due to recent criticism (Dubossarsky et al., 2018).

3.2.3 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the task of determining the senses of words in a passage of text as defined by some sense inventory, such as WordNet. Most recent work in WSD adopts the framework of Raganato et al. (2017a), which provides a standardised method for evaluating WSD systems on five datasets (Edmonds and Cotton, 2001; Snyder and Palmer, 2004; Pradhan et al., 2007; Navigli et al., 2013; Moro and Navigli, 2015) with respect to the WordNet 3.0 inventory.

WSD methods generally belong to one of two categories: supervised and knowledge-based. Supervised methods exploit sense-annotated corpora for training, whereas knowledge-based methods exploit the information contained in the sense inventory, such as WordNet's semantic relations, or some other lexical resource. Supervised methods consistently outperform knowledge-based methods by a large margin, but have the disadvantage that sense-annotated corpora are very time-consuming to produce and are available for very few languages.

While knowledge-based sense embeddings have been applied to WSD previously (Vial et al., 2017), unsupervised sense embeddings are not directly applicable to WSD. This is because there is no obvious way of mapping the set of senses learned by an unsupervised method to those defined in the reference sense inventory for a WSD task. However I will show how to perform such a mapping for the sense embedding model developed in Chapter 7.

3.2.4 Word Sense Induction

The term "Word Sense Induction" (WSI) describes precisely the task which unsupervised sense embedding systems attempt to solve: the discovery of word senses from unlabelled data, and the detection of which of these senses applies in a given context. WSI is like WSD except in that WSI systems may label word senses according to their own, internal sense inventory rather than an externally defined one. Since many previous sense embedding models have been tested on WSI (Qiu et al., 2016; Song et al., 2016; Bartunov et al., 2016; Arora et al., 2018), it is the most appropriate task for evaluating the methods in this thesis.

Two standard WSI datasets are SemEval-2010 Task 14 (Manandhar et al., 2010) and SemEval-2013 Task 13 (Jurgens and Klapaftis, 2013). Both consist of passages containing one of a set of polysemous focus words. The occurrences of the focus words in the test set are assigned gold-standard (GS) sense labels by human annotators according to a reference sense inventory. Systems may label the focus words using an arbitrary set of sense labels. Performance is evaluated using metrics which measure the degree to which the system and GS labellings are consistent.

In the SemEval-2010 dataset, each instance is labelled with a single sense, whereas in the SemEval-2013 dataset an instance may be labelled with several relevant senses, each with a corresponding weight denoting its degree of applicability in the context.

The performance metrics for SemEval-2010 are *paired F-Score* (F-S) and *V-Measure* (V-M) (Rosenberg and Hirschberg, 2007). Paired F-Score measures how often a pair of instances of a particular focus word are consistently labelled in the system and gold-standard labellings; that is, how often they are assigned the same sense labels if they have the same gold labels, or different sense labels if they have different gold labels. V-Measure "assesses the quality of a clustering solution by explicitly measuring its *homogeneity* and its *completeness*. Homogeneity refers to the degree that each cluster consists of data points primarily belonging to a single GS class, while completeness refers to the degree that each GS class consists of data points primarily assigned to a single cluster." (Manandhar et al., 2010).

The performance metrics for SemEval-2013 are Fuzzy B-Cubed (FBC) and Fuzzy Normalized Mutual Information (FNMI). These metrics take into account the fact that each example may be assigned to several senses with different weights. Generally, the F-S and FBC metrics favour systems which learn a small number of clearly delineated senses per word, while V-M and FNMI favour those which learn a large number of senses whose distinctions may be more nuanced. For this reason, overall performance on each task is typically defined as the geometric mean of its two sub-metrics.

3.2.5 Word-in-Context

Word-in-Context, or WiC (Pilehvar and Camacho-Collados, 2019), is a binary classification task in which each instance consists of two sentences containing an occurrence of a given focus word. An instance is labelled as positive if the focus word has the same sense in both sentences, and false otherwise.

As WiC is a recent dataset, sense embedding approaches have not yet been applied to it. However I will show in Chapter 7 that the techniques in this thesis can also be easily applied to WiC.

3.3 Previous Approaches

3.3.1 Clustering Methods

One of the first works in unsupervised learning of sense representations was by Schütze (1998), who proposed a two-step process, where vector representations are first derived for each context containing an ambiguous word, and these are then clustered into a pre-defined number of groups.

Huang et al. (2012) adopted this process and added a third step. They use a prediction-based neural network technique with ranking loss first introduced by Collobert and Weston (2008) to learn word embeddings. Then each word in the corpus is assigned a "contextualised" vector, calculated as a weighted sum of the

vectors in its context. The context is defined as a window of width 5 on either side, and the weighting scheme is inverse document frequency. The contextualised vectors for each word in the vocabulary are then clustered using spherical k-means (Dhillon and Modha, 2001). The third step is to re-label each word in the corpus according to its cluster, and apply the embedding method again to obtain sense rather than word representations.

Huang et al. showed that their method learned embeddings which outperformed those learned by Collobert and Weston (2008)'s on the WordSim-353 (Finkelstein et al., 2001) and SCWS word similarity datasets.

3.3.2 Joint Training Methods

A number of later approaches employed a joint training approach, where sense labelling and sense representation learning occur in parallel. Neelakantan et al. (2014), Li and Jurafsky (2015) and Bartunov et al. (2016) each proposed multi-sense variants of the Skip-gram model (Mikolov et al., 2013a,c).

In the Multi-Sense Skip-gram (MSSG) model of Neelakantan et al. (2014), each word w in the vocabulary is associated with a global vector $v_g(w)$ and K sense embeddings $v_s(w,1)$, $v_s(w,2)$, ..., $v_s(w,K)$. Each sense embedding $v_s(w,k)$ also has an associated *cluster center* $\mu(w,k)$. The context of a center word is defined to be the words in a window of width R around it, and a context vector is obtained by taking the mean of the global vectors of the words in the context. In other words, for word w_t, the context is $c_t = \{w_{t-R}, ..., w_{t-1}, w_{t+1}, ..., w_{t+R}\}$, and the context vector is $v_c(c_t) = \frac{1}{2R}\sum_{c \in c_t} v_g(c)$. The center word is assigned a cluster/sense label as follows:

$$s_t = \underset{k=1,2,...,K}{\text{argmax}}\ \text{sim}(\mu(w_t,k), v_c(c_t)), \tag{3.2}$$

where "sim" denotes cosine similarity. A standard Skip-gram update is applied to the global vectors of the context words c_t and the appropriate sense vector $v_s(w_t,s_t)$ of the center word. The cluster center $\mu(w_t,s_t)$ is updated such that it is the centroid of the context vectors of the center words so far labelled as having sense s_t.

Neelakantan et al. (2014) also proposed Non-Parametric Multi-Sense Skip-gram (NP-MSSG), a variant of MSSG where, rather than having a fixed number of senses per word, the model detects and learns new meanings of each word as required. Initially each word has a single sense. A new sense is added whenever the greatest similarity between the context vector and any of the center word's sense vectors is less than a certain threshold λ, i.e. none of the existing senses seems to fit the context.

Due to the efficiency of the Skip-gram model, the MSSG and NP-MSSG models can be trained much faster than the model of Huang et al. (2012). Neelakantan et al. showed that both their models significantly outperformed Huang et al.'s on both the WordSim-353 and SCWS datasets with much less training time. However NP-MSSG did not outperform MSSG on either task despite having the ability to learn a variable number of senses per word.

Li and Jurafsky (2015) built on the work of Huang et al. (2012) and Neelakantan et al. (2014), proposing the use of Chinese Restaurant Processes (Pitman, 1995; Griffiths et al., 2004; Teh et al., 2006) to dynamically create new word senses when none of the existing ones seem appropriate, while Bartunov et al. used non-parametric Bayesian techniques to automatically learn an appropriate number of senses for each word.

Many joint training approaches have the disadvantage that they create ambiguity in the context representation by representing context words with word embeddings in order to avoid considering the exponential number of possible sense labellings for the context. Qiu et al. (2016) and Lee and Chen (2017) propose purely sense-based approaches which can sense-label the input efficiently.

3.3.3 Other Approaches

Arora et al. (2018) took a novel approach to the problem of learning word senses. They first present a theoretical model of discourse which they use to predict that the word embedding learned by techniques such as word2vec and GloVe for a polysemous word should be very close to a linear combination of the hypothetical embed-

dings corresponding to its individual senses. They then demonstrate experimentally that this is true, and go on to propose a method for recovering the underlying sense embeddings from a pre-trained word embedding model. They evaluate the resulting sense embeddings on WSI.

While contextualisation techniques have been applied to a number of word-sense related tasks including word sense disambiguation (Huang et al., 2019; Vial et al., 2019) and induction (Amrami and Goldberg, 2018, 2019), none of these methods creates explicit sense embeddings. As far as I am aware, this thesis constitutes the first attempt to use contextualisation techniques to learn sense embeddings.

Chapter 4

Contextualised Models

4.1 A Deeper Form of Transfer Learning

The use of word embeddings can be viewed as a form of transfer learning: knowledge about words, learned through training on a task such as language modelling, is distilled in the form of vectors of real numbers. These can then be used as the input layer of a neural network trained to perform some other NLP task, generally yielding a significant gain in performance over randomly initialised weights. However, this is a shallow form of transfer learning. Language is far more complex than individual words, with many rich features such as compositionality, anaphora, long-term dependencies and negation, which word embeddings are unable to capture. Contextualised models are deep models which aim to learn transferable linguistic information at the sentence level or higher. They are typically trained on language modelling or some related unsupervised task.

It will be useful to define at this point the concept of a *contextualiser*. Most contextualised models have a contextualiser at their core. A contextualiser C is a function, typically a deep neural network of some variety, which maps a set of input representations (perhaps word embeddings) $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathbb{R}^d$ to a corresponding set of output representations $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n \in \mathbb{R}^d$, for any $n \geq 1$.

There are two main varieties of contextualisers which have been employed in contextualised NLP models so far: recurrent models and Transformer models. An

overview of both is given below.

4.2 Recurrent Models

Recurrent neural networks (RNNs) (Elman, 1990), and especially their Long Short-Term Memory (LSTM) variant (Hochreiter and Schmidhuber, 1997), have been one of the most powerful tools in the deep learning arsenal for NLP. They have been applied to speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013), machine translation (Bahdanau et al., 2014), language modelling (Sundermeyer et al., 2012), question answering (Seo et al., 2016) and many other tasks.

One of the first models used for deep transfer learning in NLP was *context2vec* (Melamud et al., 2016), a bidirectional LSTM-based model which learns context representations through training on a language modelling variant with negative sampling objective. It was shown that these representations could be used to obtain reasonable results on a WSD task.

The first contextualised model to demonstrate transferability to a wide range of tasks was ELMo (Peters et al., 2018). ELMo can be thought of as having three components, an input layer, a pair of contextualisers and an output embedding matrix. Unlike the models we have considered so far, ELMo uses *character* rather than word embeddings. The embedding for each word is a function of the embeddings of the characters it contains. We write the embedding of a word w as $\boldsymbol{x}(w) \in \mathbb{R}^d$. The contextualisers $\overrightarrow{C}$ and $\overleftarrow{C}$ are two independent 2-layer LSTMs, one used for language modelling in the forward direction and one in the backward direction, i.e. for predicting the word which precedes a sequence. Given a sequence of input words $w_1, w_2, ..., w_n$, ELMo calculates output representations

$$\overrightarrow{\boldsymbol{y}_1}, \overrightarrow{\boldsymbol{y}_2}, ..., \overrightarrow{\boldsymbol{y}_n} = \overrightarrow{C}\big(\boldsymbol{x}(w_1), \boldsymbol{x}(w_2), ..., \boldsymbol{x}(w_n)\big) \tag{4.1}$$

$$\overleftarrow{\boldsymbol{y}_n}, \overleftarrow{\boldsymbol{y}}_{n-1}, ..., \overleftarrow{\boldsymbol{y}_1} = \overleftarrow{C}\big(\boldsymbol{x}(w_n), \boldsymbol{x}(w_{n-1}), ..., \boldsymbol{x}(w_1)\big). \tag{4.2}$$

ELMo then uses a standard method in neural language modelling for estimating

probabilities of the next/previous word:

$$\overrightarrow{\mathbb{P}}(w_t = w \mid w_1, w_2, ..., w_{t-1}) = [\text{softmax}(E\overrightarrow{\boldsymbol{y}}_{t-1})]_w \tag{4.3}$$

$$\overleftarrow{\mathbb{P}}(w_t = w \mid w_n, w_{n-1}, ..., w_{t+1}) = [\text{softmax}(E\overleftarrow{\boldsymbol{y}}_{t+1})]_w, \tag{4.4}$$

where $E \in \mathbb{R}^{|V| \times d}$ is the "output embedding matrix." ELMo jointly maximises the log likelihood of the training corpus in the forward and backward directions.

Peters et al. found that once ELMo has been trained, the hidden states of the contextualiser LSTMs when it is evaluated on a given input contain transferable information about the input text. By concatenating these hidden states into a single vector and substituting this vector for the word embeddings used by previous task-specific models, they were able to obtain relative error reductions of 6-20% on tasks in question answering, textual entailment, semantic role labelling, coreference resolution, named entity extraction and sentiment analysis. This suggested that ELMo had been effective in learning transferable linguistic knowledge at the level of sentences rather than just words.

4.3 Transformer Models

4.3.1 Transformer Architecture

The Transformer is a deep learning component for sequence processing introduced by Vaswani et al. (2017). The Transformer addresses two shortcomings of the previously predominant RNN-based models for NLP. RNNs perform long chains of operations, with one expensive step per item in the input sequence. The sequential nature of these operations makes them relatively inefficient to run on modern GPU hardware. Long chains of dependencies also make them susceptible to "vanishing gradient" problems (Bengio et al., 1994), which make it difficult to learn relationships between distant items in the sequence.

The encoder component of the Transformer, which has been used in contextualised models, addresses these problems using a technique called "self-attention." Attention is a system through which a deep learning model determines which of a

number of inputs are relevant in a given context. Attention can be thought of as searching a number of keys and returning a summary of the values associated with the keys most related to the query. In the dot-product attention system used in the Transformer, queries and keys are represented by vectors in $\mathbb{R}^k$, values are represented by vectors in $\mathbb{R}^v$, and the relatedness of a key to a query is determined by the size of their dot product.

Let $\boldsymbol{q} \in \mathbb{R}^k$ be a query, $K \in \mathbb{R}^{m \times k}$ be the matrix of m keys, and $V \in \mathbb{R}^{m \times v}$ be the matrix of m values. Then the relatedness vector $\boldsymbol{r}(\boldsymbol{q}, K)$ is given by

$$\boldsymbol{r}(\boldsymbol{q}, K) = \text{softmax}(K\boldsymbol{q}) \tag{4.5}$$

and the attention output $\boldsymbol{a}(\boldsymbol{q}, K, V)$ is given by

$$\boldsymbol{a}(\boldsymbol{q}, K, V) = \boldsymbol{r}(\boldsymbol{q}, K)^\top V. \tag{4.6}$$

The Transformer uses attention with multiple "heads," each of which pays attention to a different aspect of the input by first applying its own linear projection to the query, keys and values.

In self-attention, the queries, keys and values are all the same: in the first attention layer, each embedding in the input sequence attends to every other. Multiple such self-attention layers are stacked together, with shallow feed-forward networks in between. A Transformer encoder can be thought of as a contextualiser whose output is the output of the last such layer.

The use of self-attention allows arbitrarily distant words in the sequence to interact directly with each other, alleviating the vanishing gradient problem, and attention computations are very amenable to GPU parallelisation.

4.3.2 BERT

Though OpenAI's earlier GPT model (Radford et al., 2018) had already demonstrated successful use of a Transformer-based language model for transfer learning, BERT (Devlin et al., 2019) has perhaps been the most influential Transformer-based model for transfer learning, with its use being almost ubiquitous in state-of-the-art NLP systems.

4.3.2.1 Masked Language Modelling

One significant innovation introduced in the BERT model is the "masked language modelling" (MLM) task. It has been common to incorporate *bidirectional* context when calculating a representation for a word, that is, to use context on both the left and right side. However, the ELMo model for example is bidirectional only in a shallow sense, because the forward and backward LSTMs are almost entirely independent, with their representations being concatenated only at the output. This is necessary, because they could not interact without reading the exact word which they are trying to predict. Consider the sentence

Why did the *chicken* cross the road?

When ELMo is attempting to predict the italicised word *chicken*, the forward LSTM may read only the words *Why did the*, while the backward LSTM may only read *cross the road?*, because reading further would reveal the answer. Although the bidirectionality is shallow, this approach is very efficient because a single pass of the two LSTMs over the sequence yielding states $\overrightarrow{\boldsymbol{y}_1}, \overrightarrow{\boldsymbol{y}_2}, ..., \overrightarrow{\boldsymbol{y}_n}$ and $\overleftarrow{\boldsymbol{y}_n}, \overleftarrow{\boldsymbol{y}}_{n-1}, ..., \overleftarrow{\boldsymbol{y}_1}$ is enough to produce predictions for every word in the sequence.

In the simplest version of MLM, a random subset T of the words in the sequence are replaced with a special [MASK] token and are used as targets for prediction. In other words, given an input sequence $w_1, w_2, ..., w_n$, targets $T \subseteq \{1, 2, ..., n\}$ are chosen, and the masked sequence $w'_1, w'_2, ..., w'_n$ is constructed, where $w'_t =$ [MASK] if $t \in T$, and $w'_t = w_t$ otherwise. In BERT, each word has a 15% chance of being used as a target. The objective is to maximise the estimated log-likelihood of the targets, i.e.

$$\frac{1}{|T|} \sum_{t \in T} \log \hat{\mathbb{P}}(w_t \mid w'_1, w'_2, ..., w'_n). \tag{4.7}$$

Since the target words are replaced by [MASK] and so do not appear in the input sequence, the entire context can be read by the contextualiser without the targets being exposed. This allows deep bidirectionality, in comparison with GPT, for example, which is unidirectional. However MLM is less efficient than ELMo's implementa-

tion of bidirectionality in that it learns from only $|T|$ predictions per input sequence whereas ELMo makes a prediction for every word in the sequence.

There is an extra subtlety to the MLM policy used to train BERT. Rather than every word in T begin replaced with [MASK], only 80% are. 10% are left unchanged, and 10% are replaced with a random word. In other words, for each index t,

$$\begin{cases} w'_t = w_t & \text{if } t \notin T, \\ w'_t = \text{[MASK]} & \text{with probability } 0.8 \text{ if } t \in T, \\ w'_t = w_t & \text{with probability } 0.1 \text{ if } t \in T, \\ w'_t = \text{ random word in } V & \text{with probability } 0.1 \text{ if } t \in T. \end{cases} \tag{4.8}$$

The purpose of this policy is to teach the model to produce meaningful contextualised representations when the [MASK] token does *not* appear, as it never does in the other tasks to which BERT might be applied.

4.3.2.2 Tokenisation

BERT uses WordPiece tokenisation (Wu et al., 2016), where words are broken down into commonly occurring sub-word units. For example, the word *flightless* is represented as the two tokens *flight* and *##less*, where ## denotes that the current token is a continuation of the previous word. This enables all possible words to be represented using a relatively small vocabulary, and exploits the fact that language has meaningful sub-word units (*morphemes*), such as *-less*. This allows the model to understand the commonalities between words such as *flightless*, *thoughtless* and *useless* in a way which word embedding models can not. Each of the $|V|$ word pieces in the vocabulary is assigned an embedding, yielding an embedding matrix $E \in \mathbb{R}^{|V| \times d}$.

4.3.2.3 Contextualiser

BERT uses a Transformer encoder as its contextualiser with only a few minor changes to the original implementation of Vaswani et al. (2017). Note that the

Transformer architecture as described in Section 4.3.1 does not take account of the order of the words in the sequence, as self-attention takes no account of this. Word order is obviously an extremely important aspect of language, and there are various ways of taking it into account in a Transformer model. BERT does so by learning special "positional embeddings" $\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_N \in \mathbb{R}^d$ for each index up to some maximum sequence length N, which are added to the input embeddings to provide word-order information. After splitting the input into word pieces $w_1, w_2, ..., w_n$, the position-augmented embeddings $\boldsymbol{x}_1 = \boldsymbol{e}_{w_1} + \boldsymbol{p}_1, \boldsymbol{x}_2 = \boldsymbol{e}_{w_2} + \boldsymbol{p}_2, ..., \boldsymbol{x}_n = \boldsymbol{e}_{w_n} + \boldsymbol{p}_n$ are fed into the Transformer-encoder contextualiser C, yielding corresponding output representations $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n \in \mathbb{R}^d$.

4.3.2.4 Training Regime

BERT predicts a probability distribution for the target tokens similarly to ELMo, but with the addition of a bias term $\boldsymbol{b} \in \mathbb{R}^{|V|}$:

$$\hat{\mathbb{P}}(w_t = w) = [\text{softmax}(E\boldsymbol{y}_t + \boldsymbol{b})]_w. \tag{4.9}$$

In addition to MLM, BERT is jointly trained on a second unsupervised task, *next sentence prediction* (NSP). NSP is the binary classification task of determining whether one sentence follows another in a passage of text. The purpose of this task is to encourage BERT to learn to understand relationships between sentences, something which is not well taught by language modelling. It can be considered an unsupervised task since it is trivial to generate examples from an unannotated corpus.

Each training example for BERT begins with a special [CLS] token, followed by a pair of sentences separated by a special [SEP] token, corresponding to the two sentences under consideration for NSP. The [CLS] token has no meaning of its own; the only way in which it is used is for its output representation $\boldsymbol{y}_1$. This is used as an aggregate representation of the sequence for NSP and other downstream tasks to which BERT is applied. For NSP, the aggregate representation is fed into a shallow, feed-forward network whose output is the predicted probability that the two

sentences follow each other in the corpus as opposed to being a randomly selected pair.

4.3.2.5 Fine-tuning

When ELMo is used for transfer learning, its weights are fixed and its hidden states are evaluated and used as the inputs for the target task. BERT on the other hand is designed to be "fine-tuned," where the model is adapted by adjusting its weights through training on the target task, often with the addition of an output layer. For instance, for the Multi-Genre Natural Language Inference (MNLI) task (Williams et al., 2018), in which the system must determine whether a sentence *contradicts*, *entails* or is *neutral* with respect to another, a softmax layer with outputs corresponding to the three classes is added on top of the output representation of the [CLS] token. The entire network is then trained on examples from the MNLI dataset, where the two sentences are separated by the [SEP] token.

4.3.2.6 Impact

The $\mathbf{BERT}_{\mathbf{LARGE}}$ model produced by Devlin et al. (2019) achieved state-of-the-art performance on all 8 tasks of the GLUE natural language understanding benchmark (Wang et al., 2018), on many by a large margin. BERT has been used in many state-of-the-art systems for various tasks, including word sense induction (Amrami and Goldberg, 2019) and disambiguation (Huang et al., 2019), coreference resolution (Joshi et al., 2019) and sentiment analysis (Sun et al., 2019).

Chapter 5

A Skip-gram-Inspired Proof of Concept

5.1 Motivation

Recall from Section 1.4.1 the idea which underlies this thesis, namely that the probability of a word occurring in a given context is equal to the sum of the probabilities of its individual senses occurring:

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \mathbb{P}(s \mid c), \tag{5.1}$$

where S_w is the set of senses for word $w \in V$. For instance, intuitively we might have $S_{\text{rock}} = \{\text{rock:stone, rock:musical genre, rock:shake}\}$. We assume that the S_w are disjoint, i.e. $S_w \cap S_{w'} = \emptyset$ whenever $w \neq w'$, and we define the full sense inventory $S = \bigcup_{w \in V} S_w$.

It has been argued in Section 1.4.1 that there is good theoretical justification for the idea expressed by Equation 5.1, but now it must be shown that it can be used as the basis of a practical method for learning sense embeddings.

An obvious starting point is the Skip-gram model of Mikolov et al. (2013a,c) due to its simplicity, efficiency and effectiveness. Recall that the Skip-gram method learns to discriminate between words which tend to appear in the context of a given

centre word c and those which do not by maximising the value of

$$\sigma(\boldsymbol{u}_c^\top \boldsymbol{v}_o), \tag{5.2}$$

which corresponds loosely to the probability of words c and o co-occurring, when c and o actually appear in the same context in the training corpus, and minimising the same expression when o is a randomly sampled word which does not appear in c's context.

5.2 Formulation

5.2.1 Probability of word given context

The Skip-gram method defines the context words as those contained within a window of width m around the centre word. Thus Equation 5.1 can be rewritten as

$$\mathbb{P}(w_t \mid c) \equiv \mathbb{P}(w_t \mid w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}) \tag{5.3}$$

$$= \sum_{s \in S_{w_t}} \mathbb{P}(s \mid w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}). \tag{5.4}$$

In order to be able to apply the Skip-gram method, we rearrange our equation using Bayes' theorem to obtain an expression involving probability of context given the sense of the centre word rather than vice versa:

$$\mathbb{P}(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s \mid w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}) \tag{5.5}$$

$$= \sum_{s \in S_{w_t}} \frac{\mathbb{P}(w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m} \mid s)\, \mathbb{P}(s)}{\mathbb{P}(w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m})}. \tag{5.6}$$

Treating each context word independently, we have

$$\mathbb{P}(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s) \prod_{i=t-m, \neq t}^{t+m} \frac{\mathbb{P}(w_i \mid s)}{\mathbb{P}(w_i)} \tag{5.7}$$

$$= \sum_{s \in S_{w_t}} \mathbb{P}(s) \prod_{i=t-m, \neq t}^{t+m} f(w_i, s), \tag{5.8}$$

where $f(w_i, s)$ denotes the ratio between the probability of w_i occurring in the context of word sense s, and of w_i occurring in the corpus as a whole, i.e. its

unigram probability. In other words, $f(w_i, s)$ estimates how many times more (or less) likely w_i is to occur when s is present. Since $f(w_i, s)$ must be positive, but may be greater than 1, a natural choice of function to represent f is

$$f(w, s) = e^{\boldsymbol{u}_s^\top \boldsymbol{v}_w}, \tag{5.9}$$

where $\boldsymbol{u}_s \in \mathbb{R}^d$ denotes the sense embedding for sense s and $\boldsymbol{v}_w \in \mathbb{R}^d$ denotes the word embedding for word w. Thus we have

$$\mathbb{P}(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s) \prod_{i=t-m, \neq t}^{t+m} e^{\boldsymbol{u}_s^\top \boldsymbol{v}_{w_i}} \tag{5.10}$$

$$= \sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\sum_{i=t-m, \neq t}^{t+m} \boldsymbol{u}_s^\top \boldsymbol{v}_{w_i}}. \tag{5.11}$$

Because the exponent in the above expression is potentially large, we scale it by $\frac{1}{2m}$ to avoid numerical issues, yielding

$$\mathbb{P}(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\frac{1}{2m} \sum_{i=t-m, \neq t}^{t+m} \boldsymbol{u}_s^\top \boldsymbol{v}_{w_i}} \tag{5.12}$$

$$= \sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \left(\frac{1}{2m} \sum_{i=t-m, \neq t}^{t+m} \boldsymbol{v}_{w_i} \right)}. \tag{5.13}$$

Note that the term $\frac{1}{2m} \sum_{i=t-m, \neq t}^{t+m} \boldsymbol{v}_{w_i}$ is simply the mean of the embeddings of the context words, for which we will now write $\boldsymbol{c}$:

$$\mathbb{P}(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \boldsymbol{c}}. \tag{5.14}$$

5.2.2 "Unigram" sense probability

$\mathbb{P}(s)$, the "unigram" probability of an individual word sense, is unknown and must be learned as a parameter of the model. One way to do this would be to express $\mathbb{P}(s)$ as an output of the softmax function over the entire sense inventory:

$$\mathbb{P}(s) = \frac{e^{l_s}}{\sum_{s' \in S} e^{l_{s'}}}, \tag{5.15}$$

where $\boldsymbol{l} \in \mathbb{R}^{|S|}$. However, all $|S|$ parameters of $\boldsymbol{l}$ would need to be updated for every training example, significantly worsening the time complexity. Instead, $\mathbb{P}(s)$ can be broken down into two components, $\mathbb{P}(s \mid w)$ and $\mathbb{P}(w)$:

$$\mathbb{P}(s) = \mathbb{P}(s \mid w)\, \mathbb{P}(w), \tag{5.16}$$

where w is the word of which s is a sense. $\mathbb{P}(w)$, the unigram probability of word w, is known. $\mathbb{P}(s \mid w)$, which can be read as "how often word w has sense s" is unknown, and can be learned as a softmax output:

$$\mathbb{P}(s \mid w) = \frac{e^{l_{ws}}}{\sum_{s' \in S_w} e^{l_{ws'}}}, \tag{5.17}$$

for $\boldsymbol{l}_w \in \mathbb{R}^{|S_w|}$ for all $w \in V$. This expression contains only $|S_w|$ parameters, and since $|S_w| << |S|$, evaluating it and updating the parameters does not represent a significant cost. Our final expression for $\mathbb{P}(s)$ is therefore

$$\mathbb{P}(s) = p_w \frac{e^{l_{ws}}}{\sum_{s' \in S_w} e^{l_{ws'}}}, \tag{5.18}$$

where $p_w = \mathbb{P}(w)$ is the unigram probability of word w.

5.2.3 Objective function

In the Skip-gram model, the expression $\sigma(\boldsymbol{u}_c^\top \boldsymbol{v}_o)$ loosely corresponds to the probability of word o appearing in the context of centre word c. However, there is no component of the model which ensures that $\mathbb{P}(o \mid c)$ is actually a valid probability distribution, i.e. that

$$\sum_{o \in V} \sigma(\boldsymbol{u}_c^\top \boldsymbol{v}_o) = 1. \tag{5.19}$$

This is not a problem for the Skip-gram model because it does not affect the quality of the word embeddings learned. Our sense embedding model however relies on the property

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \mathbb{P}(s \mid c), \tag{5.20}$$

so we need to ensure that $\mathbb{P}(w \mid c)$ approximates a genuine probability distribution. Noise-contrastive estimation (NCE) (Gutmann and Hyvärinen, 2012), upon which the Skip-gram method's negative sampling is based, offers a way to do so. NCE has previously been applied to language modelling by Mnih and Teh (2012). As with negative sampling, the objective is to learn to discriminate between examples drawn from a true, "data" distribution $\mathbb{P}_d$ and a "noise" distribution $\mathbb{P}_n$. In our case, these

distributions are over the centre word w given context c. If samples from the noise distribution are k times more frequent than samples from the data distribution, then given a context c, the mixed probability $\mathbb{P}_m(w \mid c)$ of a given centre word occurring can be expressed as

$$\mathbb{P}_m(w \mid c) = \frac{1}{k+1}\mathbb{P}_d(w \mid c) + \frac{k}{k+1}\mathbb{P}_n(w \mid c). \tag{5.21}$$

Thus if in a training example, centre word w occurs in a context c, the probability that w is drawn from the data distribution (denoted by $D = 1$) can be expressed as

$$\mathbb{P}(D = 1 \mid w, c) = \frac{\mathbb{P}_d(w \mid c)}{\mathbb{P}_m(w \mid c)} \tag{5.22}$$

$$= \frac{\frac{1}{k+1}\mathbb{P}_d(w \mid c)}{\frac{1}{k+1}\mathbb{P}_d(w \mid c) + \frac{k}{k+1}\mathbb{P}_n(w \mid c)} \tag{5.23}$$

$$= \frac{\mathbb{P}_d(w \mid c)}{\mathbb{P}_d(w \mid c) + k\mathbb{P}_n(w \mid c)}. \tag{5.24}$$

Similarly, the probability that w is drawn from the noise distribution (D = 0) is given by

$$\mathbb{P}(D = 0 \mid w, c) = 1 - \mathbb{P}(D = 1 \mid w, c) \tag{5.25}$$

$$= 1 - \frac{\mathbb{P}_d(w \mid c)}{\mathbb{P}_d(w \mid c) + k\mathbb{P}_n(w \mid c)} \tag{5.26}$$

$$= \frac{k\mathbb{P}_n(w \mid c)}{\mathbb{P}_d(w \mid c) + k\mathbb{P}_n(w \mid c)}. \tag{5.27}$$

Consider a context $c = w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}$ for which we know the true centre word w_t, and suppose we draw k fake centre words $w'_1, w'_2, ..., w'_k$ from the noise distribution. Then the estimated likelihood of the true and fake centre words belonging to their respective classes is given by

$$\mathcal{L} = \mathbb{P}(D = 1 \mid w_t, c) \prod_{i=1}^{k} \mathbb{P}(D = 0 \mid w'_i, c). \tag{5.28}$$

We maximise the log-likelihood $J(w_t, c\,; \boldsymbol{u}_{:}, \boldsymbol{v}_{:}, \boldsymbol{l}_{:})$:

$$J(w_t, c\,; \boldsymbol{u}_{:}, \boldsymbol{v}_{:}, \boldsymbol{l}_{:}) = \log \mathbb{P}(D = 1 \mid w_t, c) + \sum_{i=1}^{k} \log \mathbb{P}(D = 0 \mid w'_i, c) \tag{5.29}$$

$$= \log \frac{\mathbb{P}_d(w_t \mid c)}{\mathbb{P}_d(w_t \mid c) + k\mathbb{P}_n(w_t \mid c)} + \sum_{i=1}^{k} \log \frac{k\mathbb{P}_n(w'_i \mid c)}{\mathbb{P}_d(w'_i \mid c) + k\mathbb{P}_n(w'_i \mid c)}. \tag{5.30}$$

Recall that we derived the expression

$$\mathbb{P}_d(w_t \mid c) = \sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \boldsymbol{c}} \tag{5.31}$$

for the data distribution, and we follow Mikolov et al. (2013c) in using the expression

$$P_n(w) = \frac{p_w^{3/4}}{\sum_{v \in V} p_v^{3/4}} \tag{5.32}$$

for the noise distribution. Note that this expression is not dependent on the context, so we can write $\mathbb{P}_n(w \mid c)$ as $\mathbb{P}_n(w)$ from now on. Now we have

$$\begin{aligned} J(w_t, c\,; \boldsymbol{u}_{:}, \boldsymbol{v}_{:}, \boldsymbol{l}_{:}) = \log \frac{\sum_{s \in S_{w_t}} \mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \boldsymbol{c}}}{\sum_{s \in S_{w_t}} \left(\mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \boldsymbol{c}}\right) + k\mathbb{P}_n(w_t)} + \\ \sum_{i=1}^{k} \log \frac{k\mathbb{P}_n(w_i')}{\sum_{s \in S_{w_i'}} \left(\mathbb{P}(s)\, e^{\boldsymbol{u}_s^\top \boldsymbol{c}}\right) + k\mathbb{P}_n(w_i')}, \end{aligned} \tag{5.33}$$

where $\boldsymbol{c} = \frac{1}{2m} \sum_{i=t-m, \neq t}^{t+m} \boldsymbol{v}_{w_i}$ is the mean of the context vectors and $\mathbb{P}(s)$ is as described in Section 5.2.2:

$$\mathbb{P}(s) = p_w \frac{e^{l_{ws}}}{\sum_{s' \in S_w} e^{l_{ws'}}}. \tag{5.34}$$

5.3 Implementation Details

5.3.1 Corpus

Like BERT, we use a training corpus consisting of plain text extracted from a recent English Wikipedia dump, and text from BookCorpus (Zhu et al., 2015), which consists of a large number of books written in English. Wikipedia contains high quality formal writing on a wide range of topics. BookCorpus is complementary in that it contains some types of language which are infrequent in Wikipedia, such as informal language, direct speech and narrative.

5.3.2 Preprocessing

The corpus was first tokenised using Stanford CoreNLP's tokeniser (Manning et al., 2014). The corpus contained approximately 3.6 billion tokens in total. All tokens

were then converted to lower case. The vocabulary was chosen to be the ~86K tokens appearing more than 500 times in the corpus. All other tokens were replaced with a special [UNK] token.

5.3.3 Parameters

We use a word and sense embedding dimensionality of $d = 256$. To keep the total number of model parameters reasonable, we allow only the ~10,000 words which appear more than 20,000 times in the training corpus to have multiple sense embeddings, and we set the number of sense embeddings for these words to be 5. Other words are assigned only a single sense embedding. The justification for doing so is that according to Zipf's law (Zipf, 1950), infrequent words are much less likely to have multiple meanings.

5.3.4 Initialisation

Each element of the word and sense embeddings $\boldsymbol{u}$ and $\boldsymbol{v}$ is initially set to a value drawn at random from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \frac{1}{\sqrt{d}}$.

5.4 Results

To evaluate the results, we visualise the nearest neighbours of the senses of a number of polysemous words. Table 5.1 shows the nearest neighbours according to cosine similarity of the 5 sense embeddings learned for the words *bank*, *cell*, *left*, *apple* and *fox*. It is comparable to Table 3.1, which shows similar results for previous sense embedding methods.

The Skip-gram inspired method has learned embeddings which appear to correspond to at least two of the senses of the words of interest. Since most of these words have fewer than 5 meanings, we see that there are often several embeddings per word which seem to correspond to the same meaning. It is encouraging to see that this method appears to have detected a meaning of *fox* which was missed by the

Sense	Nearest Neighbours
$bank_1$	$bank_5$, $banks_3$, $riverbank_1$, $tiber_1$, $elbe_1$, $tributary_4$
$bank_2$	$banks_1$, $bank_4$, $banks_4$, $citibank_1$, $hsbc_1$, $mortgage_5$
$bank_3$	$icici_1$, $subtrust_1$, $exim_1$, sbi_1, $bank_4$, $grameen_1$
$bank_4$	$account_4$, $banks_1$, $accounts_5$, $bank_3$, $bank_2$, $banks_4$
$bank_5$	$bank_1$, $berm_1$, $riverbank_1$, $embankment_1$, $riverbed_1$, $driveway_5$
$cell_1$	$cells_3$, $cell_4$, $cells_5$, $cell_3$, $cells_2$, $cellular_5$
$cell_2$	$mobile_4$, $cordless_1$, $cellular_2$, $landline_1$, $rang_2$, $phone_5$
$cell_3$	$cells_2$, $cells_1$, $cells_4$, $cell_4$, $molecule_3$, $molecule_1$
$cell_4$	$cell_3$, $cells_1$, $cells_5$, $cytoplasm_1$, $membrane_3$, $cells_3$
$cell_5$	$prison_3$, $cubicle_1$, $jail_3$, $cells_4$, $jail_5$, $prison_5$
$left_1$	$left_4$, $leaving_4$, $remained_2$, $leave_2$, $lagged_1$, $leaving_3$
$left_2$	$departed_3$, $rejoined_3$, $departed_5$, $re\text{-}joined_1$, $rejoined_1$, $quit_1$
$left_3$	$tackle_4$, $halfback_1$, $wingers_1$, $overhand_1$, $center_4$, $fielder_1$
$left_4$	$leave_5$, $leaving_4$, $leaves_3$, $ditched_1$, $leaving_1$, $kept_1$
$left_5$	$right_3$, $right_5$, $left_3$, $wrist_3$, $dislocated_1$, $heel_4$
$apple_1$	$apple_4$, $vegetables_4$, $honey_2$, $tomatoes_2$, $tomatoes_3$, $vegetables_3$
$apple_2$	$apple_3$, $microsoft_1$, $microsoft_5$, $microsoft_2$, $apple_1$, $apple_4$
$apple_3$	$apple_2$, $macintosh_1$, ibm_2, $microsoft_3$, ibm_1, $microsoft_4$
$apple_4$	$apple_1$, $guava_1$, $honey_2$, $rhubarb_1$, $persimmon_1$, $artichoke_1$
$apple_5$	$apples_1$, $fruit_1$, $mango_1$, $cherry_1$, $apple_4$, $orchard_1$
fox_1	fox_4, $sullivan_2$, $knight_3$, $lippert_1$, $shaw_3$, $shaw_5$
fox_2	nbc_3, nbc_5, cbs_1, nbc_4, cbs_2, $zanuck_1$
fox_3	fox_4, $badger_1$, $squirrel_1$, $pheasant_1$, $coyote_1$, $jackal_1$
fox_4	fox_1, fox_3, $hawks_5$, $varden_1$, $hawks_1$, $waccamaw_1$
fox_5	nbc_5, nbc_2, cbs_3, abc_4, abc_2, nbc_3

Table 5.1: Nearest neighbours of sense embeddings learned by the Skip-gram based model.

methods of Neelakantan et al. (2014) and Qiu et al. (2016), namely the "surname" sense captured by fox_1.

As expected, sense embeddings which correspond to the same meaning often seem to appear in each others' lists of nearest neighbours. $bank_1$ and $bank_5$ seem to correspond to the "riverbank" sense of *bank*, while $bank_{2,3,4}$ seem to correspond to the "financial institution" sense, and indeed we find $bank_1$ is a neighbour of $bank_5$, but not of $bank_2$, $bank_3$ or $bank_4$. While the sense embedding neighbours for *cell* also seem consistent in this manner, they are not for *apple*. For instance, $apple_4$, which corresponds to the "fruit" sense, appears in the neighbour list of $apple_2$, which corresponds to the "technology company" sense. This suggests that some degree of meaning conflation is still occurring in this model: sense embeddings

corresponding to different meanings are not separating from each other completely.

Another weakness is that the model appears to have failed to find some word meanings, especially rare ones. For example, *bank* can be a verb meaning "to tilt," while *left* also refers to a political orientation, i.e. "left wing."

Overall, the results appear to confirm that the underlying hypothesis expressed in Equation 5.1 can be used as the basis of a sense embedding model. In Chapter 6, applying the same hypothesis will enable the exploitation of contextualisation techniques to develop a powerful new method for obtaining sense embeddings.

Chapter 6

Maximum Likelihood Contextual Clustering

6.1 Overview

I now present Maximum Likelihood Contextual Clustering (MLCC), a method for learning embeddings for individual word senses given a pre-trained contextualised word embedding model. To obtain sense embeddings for a given word, a number of contexts in which the word appears ("positive contexts") and does not appear ("negative contexts") are first gathered, and the contextualised model is used to obtain contextualised representations for each of these contexts. We then attempt to learn multiple embeddings for the word which allow us to discriminate as well as possible between positive and negative contexts. If the word is polysemous, we would expect the positive contexts to have representations which form several clusters in vector space, surrounded by a sea of negative contexts. Therefore we would expect the model to be able to discriminate best between positive and negative contexts when the embeddings for the word correspond roughly to the centres of these clusters.

MLCC is not a typical clustering method in that the cluster centers (i.e. sense embeddings) are both attracted towards a set of positive examples and repelled from a set of negative examples. This is because to understand the meaning of a word

sense, it is necessary to both know the contexts in which it is likely to occur, and those in which it is unlikely to occur. A statistical model of word sense which only saw the contexts in which a word does occur would conclude that it is certain to occur in any context.

6.2 Formulation

A contextualised word embedding model such as BERT can be thought of as consisting of two components, an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, and a contextualising function C which maps a sequence of input embeddings $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathbb{R}^d$ to a corresponding sequence of output representations $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n \in \mathbb{R}^d$.

The probability of token $w \in V$ occurring in a context with output representation $\boldsymbol{y}$ is given by:

$$\mathbb{P}(w \mid \boldsymbol{y}) = [\text{softmax}(E\boldsymbol{y} + \boldsymbol{b})]_w \tag{6.1}$$

$$= \frac{e^{\boldsymbol{e}_w^\top \boldsymbol{y} + b_w}}{\sum_{j \in V} e^{\boldsymbol{e}_j^\top \boldsymbol{y} + b_j}}, \tag{6.2}$$

where $\boldsymbol{e}_j$ is the jth row of E, i.e. the embedding corresponding to token j, and $\boldsymbol{b}$ is a bias vector.

Let us write the set S_w of senses of w as $S_w = \{w^{(1)}, w^{(2)}, ..., w^{(1)}_{|S_w|}\}$. Suppose we replace the embedding $\boldsymbol{e}_w$ and bias b_w for word w with embeddings $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{|S_w|}$ and biases $\beta_1, \beta_2, ..., \beta_{|S_w|}$ corresponding to its individual senses $w^{(1)}, w^{(2)}, ..., w^{(|S_w|)}$. Then, with Equation 6.2 as the starting point, the probability of sense $w^{(i)}$ occurring in a context with representation $\boldsymbol{y}$ can be written as

$$\mathbb{P}(w^{(i)} \mid \boldsymbol{y}\,; \Theta, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\theta}_i^\top \boldsymbol{y} + \beta_i}}{\sum_{j \in V \setminus \{w\}} e^{\boldsymbol{e}_j^\top \boldsymbol{y} + b_j} + \sum_{j=1}^{|S_w|} e^{\boldsymbol{\theta}_j^\top \boldsymbol{y} + \beta_j}}, \tag{6.3}$$

where Θ denotes the full set of sense embeddings $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{|S_w|}$ and $\boldsymbol{\beta}$ denotes the full set of sense biases $\beta_1, \beta_2, ..., \beta_{|S_w|}$. Note that the term $e^{\boldsymbol{e}_w^\top \boldsymbol{y} + b_w}$ in the denominator of Equation 6.2 has been replaced by the sum of the corresponding terms for w's sense embeddings, $\sum_{j=1}^{|S_w|} e^{\boldsymbol{\theta}_j^\top \boldsymbol{y} + \beta_j}$.

Recall Equation 1.1, which links the probability of a word occurring with the probability of its individual senses occurring:

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \mathbb{P}(s \mid c).$$

In this situation, the context c is represented by the contextualised embedding $\boldsymbol{y}$, so we have

$$\mathbb{P}(w \mid \boldsymbol{y}\,; \Theta, \boldsymbol{\beta}) = \sum_{i=1}^{|S_w|} \mathbb{P}(w^{(i)} \mid \boldsymbol{y}\,; \Theta, \boldsymbol{\beta}) \tag{6.4}$$

$$= \frac{\sum_{i=1}^{|S_w|} e^{\boldsymbol{\theta}_i^\top \boldsymbol{y} + \beta_i}}{\sum\limits_{j \in V \setminus \{w\}} e^{\boldsymbol{e}_j^\top \boldsymbol{y} + b_j} + \sum_{j=1}^{|S_w|} e^{\boldsymbol{\theta}_j^\top \boldsymbol{y} + \beta_j}}. \tag{6.5}$$

Now that we have an expression for the probability of w occurring in a given context in terms of its sense embeddings, we can work towards defining a learning objective for sense embeddings.

The contexts in the training corpus (perhaps the same one used to train the contextualiser) can be partitioned into two sets, C^+, those in which w occurs, and C^-, those in which it does not occur. For the purposes of MLCC, a context is defined to be a sequence of words $c = w_1, w_2, ..., w_n$ of length up to 128 (MiniBERT's maximum sequence length). If $w_t = w$ for some $t \in \{1, 2, ..., n\}$, then c belongs to C^+, otherwise it belongs to C^-. The contextualised representation $\boldsymbol{y}(c)$ is found by replacing w_t with [MASK] (where t is a random index if $c \in C^-$) and taking the contextualiser output $\boldsymbol{y}_t$ for the resulting sequence.

Since the corpus may be very large, we form a smaller dataset of positive examples D^+ by selecting a random subset of C^+, and a dataset of negative examples D^- by selecting a random subset of C^-. Thus the full training dataset is $D^+ \cup D^-$. We are likely to select a ratio $\frac{|D^+|}{|D^-|}$ of positive to negative examples for the training dataset which is higher than the ratio $\frac{|C^+|}{|C^-|}$ observed in the corpus as a whole, as $\frac{|C^+|}{|C^-|}$ is very low except when w is an extremely frequent word such as *the*.

We attempt to learn sense embeddings for w which maximise the likelihood of the observed labels of the contexts in D^+ and D^-. The estimated probability that a

context c is positive given that it occurs in the training dataset is given by

$$\mathbb{P}(c \in D^+ \mid c \in D^+ \cup D^-) = \frac{\mathbb{P}(c \in D^+ \ \wedge \ c \in D^+ \cup D^-)}{\mathbb{P}(c \in D^+ \cup D^-)} \tag{6.6}$$

$$= \frac{\mathbb{P}(c \in D^+)}{\mathbb{P}(c \in D^+) + \mathbb{P}(c \in D^-)}. \tag{6.7}$$

The desired estimated probabilities $\mathbb{P}(c \in D^+)$ and $\mathbb{P}(c \in D^-)$ that a context c drawn randomly from the training corpus is found in either the positive or negative datasets D^+ and D^- can be expressed as

$$\mathbb{P}(c \in D^+) = \mathbb{P}(c \in D^+ \mid c \in C^+)\, \mathbb{P}(c \in C^+) \tag{6.8}$$

$$= \frac{|D^+|}{|C^+|}\mathbb{P}(w \mid c), \tag{6.9}$$

$$\mathbb{P}(c \in D^-) = \mathbb{P}(c \in D^- \mid c \in C^-)\, \mathbb{P}(c \in C^-) \tag{6.10}$$

$$= \frac{|D^-|}{|C^-|}\big(1 - \mathbb{P}(w \mid c)\big), \tag{6.11}$$

where $\mathbb{P}(w \mid c)$ is shorthand for $\mathbb{P}(w \mid \boldsymbol{y}(c)\ ; \Theta, \boldsymbol{\beta})$, and is defined according to Equation 6.5. Combining these expressions with Equation 6.7, we have

$$\mathbb{P}(c \in D^+ \mid c \in D^+ \cup D^-) = \frac{\frac{|D^+|}{|C^+|}\mathbb{P}(w \mid c)}{\frac{|D^+|}{|C^+|}\mathbb{P}(w \mid c) + \frac{|D^-|}{|C^-|}\big(1 - \mathbb{P}(w \mid c)\big)} \tag{6.12}$$

$$= \frac{\mathbb{P}(w \mid c)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)}, \tag{6.13}$$

where $\rho = \frac{|C^+||D^-|}{|C^-||D^+|} = \frac{\text{ratio of positive to negative examples in the corpus}}{\text{ratio of positive to negative examples in the training dataset}}$. Thus typically $\rho << 1$. Similarly we have

$$\mathbb{P}(c \in D^- \mid c \in D^+ \cup D^-) = 1 - \mathbb{P}(c \in D^+ \mid c \in D^+ \cup D^-) \tag{6.14}$$

$$= \frac{\rho\big(1 - \mathbb{P}(w \mid c)\big)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)}. \tag{6.15}$$

We can now form an expression for the likelihood $\mathcal{L}(\Theta, \boldsymbol{\beta})$ of the labels of the training examples, parameterised by the sense embeddings Θ and biases $\boldsymbol{\beta}$:

$$\mathcal{L}(\Theta, \boldsymbol{\beta}) = \prod_{c \in D^+} \mathbb{P}(c \in D^+ \mid c \in D^+ \cup D^-) \prod_{c \in D^-} \mathbb{P}(c \in D^- \mid c \in D^+ \cup D^-) \tag{6.16}$$

$$= \prod_{c \in D^+} \frac{\mathbb{P}(w \mid c)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)} \prod_{c \in D^-} \frac{\rho\big(1 - \mathbb{P}(w \mid c)\big)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)}. \tag{6.17}$$

Recall that the estimated probability $\mathbb{P}(w \mid c)$ is parameterised by Θ and $\boldsymbol{\beta}$, but we omit this in our formulae for readability. To remove products from the expression and frame the problem with a minimisation objective $J(\Theta, \boldsymbol{\beta})$, we take the negative logarithm of $\mathcal{L}(\Theta, \boldsymbol{\beta})$:

$$J(\Theta, \boldsymbol{\beta}) = -\log \mathcal{L}(\Theta, \boldsymbol{\beta}) \tag{6.18}$$

$$= -\log \Bigg(\prod_{c \in D^+} \frac{\mathbb{P}(w \mid c)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)} \prod_{c \in D^-} \frac{\rho\big(1 - \mathbb{P}(w \mid c)\big)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)} \Bigg) \tag{6.19}$$

$$= -\sum_{c \in D^+} \log \frac{\mathbb{P}(w \mid c)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)} - \sum_{c \in D^-} \log \frac{\rho\big(1 - \mathbb{P}(w \mid c)\big)}{\mathbb{P}(w \mid c) + \rho\big(1 - \mathbb{P}(w \mid c)\big)}. \tag{6.20}$$

6.3 Improving Distinctness of Word Senses

One issue with the sense embeddings produced in Chapter 5 is that there were often two or more embeddings for the same word which appeared to correspond to the same meaning. A more desirable behaviour would be for there to be a single embedding for each word sense, and for any extra sense embeddings to "die off," that is have a very low predicted probability of occurring in any context. This raises the question of how to determine how similar or distinct the meanings of two embeddings are. A sensible definition seems to be

> "If two sense embeddings tend to have high predicted probabilities of occurring in the same contexts, then they correspond to similar meanings."

Since a word generally has only a single meaning in a given context, we would expect only one of its senses to have a high predicted probability per context. This suggests we ought to add a term to the objective function which encourages this to occur.

Recall that the estimated probability $\mathbb{P}(w^{(i)} \mid c)$ of word w occurring in context c with its ith sense is proportional to $e^{\boldsymbol{\theta}_i^\top \boldsymbol{y}(c) + \beta_i}$, where $\boldsymbol{\theta}_i$ and β_i are the sense embedding and bias respectively for $w^{(i)}$ and $\boldsymbol{y}(c)$ is the contextualised representation of

c. Therefore if we know that w does indeed occur in context c, then the probability that it has sense $w^{(i)}$ is given by

$$\mathbb{P}(w^{(i)} \mid c, w) = \frac{e^{\boldsymbol{\theta}_i^\top \boldsymbol{y}(c)+\beta_i}}{\sum_{j=1}^{|S_w|} e^{\boldsymbol{\theta}_j^\top \boldsymbol{y}(c)+\beta_j}}. \tag{6.21}$$

For each context in which w occurs (i.e. each context in D^+), we want to encourage $\mathbb{P}(w^{(i)} \mid c, w)$ to be large for one i in $\{1, 2, ..., |S_w|\}$, and small for other i. An obvious approach is to add a "distinctness loss" term $J_d(\Theta, \boldsymbol{\beta})$ to the objective which is proportional to the probability of the most likely sense in context:

$$J_d(\Theta, \boldsymbol{\beta}) = -\lambda_d \sum_{c \in D^+} \max_{i=1,...,|S_w|} \mathbb{P}(w^{(i)} \mid c, w), \tag{6.22}$$

where λ_d is a hyperparameter controlling the strength of the effect.

This approach has the disadvantage that the $\max$ function is not smooth, which can cause problems when a gradient-based optimisation method is used. Furthermore, it is likely to lead to one sense quickly acquiring all of the probability mass and preventing a wide range of senses from being learned. Ideally we would like to find a smooth function of a vector of probabilities $\boldsymbol{p}$ which is maximised when one element is large and the rest are small. One possibility is

$$f(\boldsymbol{p}) = \sum_i p_i^r, \tag{6.23}$$

where $r > 1$ is a constant. Since each p_i lies in the interval $[0, 1]$, raising p_i to a power greater than 1 decreases it, and the smaller p_i is, the greater the factor by which it is decreased. Therefore $f(\boldsymbol{p})$ is maximised at a value of 1 when one element of $\boldsymbol{p}$ is 1 and the rest are zero, and minimised when all elements of $\boldsymbol{p}$ are equal. This is exactly the behaviour we desire. For reasons which will be explored in Section 7.2.5.2, a logarithmic transformation and a coefficient of $\frac{1}{r}$ are applied to obtain the following smooth formulation of the distinctness loss:

$$J_d(\Theta, \boldsymbol{\beta}) = \frac{-1}{r} \sum_{c \in D^+} \log \sum_{i=1}^{k} \left[\mathbb{P}(w^{(i)} \mid c, w)\right]^r. \tag{6.24}$$

6.4 Experiments

The performance of MLCC is evaluated on word sense induction (WSI), described in Section 3.2.4. I first describe the contextualised word embedding model used

in the MLCC experiments, MiniBERT, then state the parameters of the clustering method and present the results.

6.4.1 MiniBERT

Chapter 7 presents a novel, end-to-end contextualised sense embedding model called PolyLM. Since the resources available were insufficient to train a model of $\textbf{BERT}_{\textbf{LARGE}}$'s size, PolyLM is significantly smaller in terms of number of parameters and training time. Wherever a contextualised word embedding model is required by way of comparison, rather than using $\textbf{BERT}_{\textbf{LARGE}}$, we instead use MiniBERT, a model trained specifically to have similar dimensions and parameters to PolyLM. MiniBERT and PolyLM both use the corpus described in Section 5.3.1, but with an additional lemmatisation step. This and additional details of MiniBERT are described below.

6.4.1.1 Lemmatisation

In linguistics, the *lemma* of a word is its un-inflected form. For instance, the lemma of *cats* is *cat* and the lemma of *going* is *go*.

Different forms of the same lemma, such as *run*, *runs* and *running*, tend to share the same set of senses (ignoring the effect the inflectional morphology *-s* and *-ing* has on the sense). In the SemEval-2010 and SemEval-2013 WSI tasks, the system is required to learn a set of senses for a given *lemma*; the test examples may contain any inflectional form of the lemma.

To enable learning senses at the lemma level, the training corpus is *lemmatised*; that is, its words are converted to lemma form. Lemmatisation is a non-trivial procedure, because many words are forms of more than one lemma. For instance, the word "saw" is both the past tense form of the lemma "see," and is itself a lemma denoting a tool for cutting. Therefore the correct lemmatisation of "saw" depends on whether it is being used as a verb or a noun.

The lemmatisation procedure applied to the corpus is as follows: first, part-of-speech (POS) tagging is performed using Stanford CoreNLP's POS tagger (Manning et al., 2014). Since the inflectional morphology is useful information for lan-

guage modelling, it is not thrown away. Instead, each word whose part of speech tag denotes that it has inflectional morphology is split into two separate tokens, its lemmatised form and a special token corresponding to the particular inflection, as shown in Table 6.1. The lemmatised form is found using NLTK's WordNetLemmatizer (Bird et al., 2009). Words with other part of speech tags are not modified.

PTB tag	**Part of speech denoted**	**Example Lemmatisation**
NNS	Noun, plural	*cats* ⇒ *cat* + [PL]
JJR	Adjective, comparative	*bigger* ⇒ *big* + [COMP]
JJS	Adjective, superlative	*biggest* ⇒ *big* + [SUP]
RBR	Adverb, comparative	*further* ⇒ *far* + [COMP]
RBS	Adverb, superlative	*furthest* ⇒ *far* + [SUP]
VBD	Verb, past tense	*saw* ⇒ *see* + [PAST]
VBG	Verb, gerund/present participle	*seeing* ⇒ *see* + [GER]
VBZ	Verb, 3rd person singular present	*sees* ⇒ *see* + [3RD]
VBP	Verb, non-3rd person singular present	*see* ⇒ *see* + [N3RD]
VBN	Verb, past participle	*seen* ⇒ *see* + [PART]

Table 6.1: Penn TreeBank (PTB) part of speech tags denoting inflectional morphology with example lemmatisations. Note that as comparatives and superlatives are effectively the same semantic operation on both adjectives and adverbs, they are denoted with the same special [COMP] and [SUP] tokens.

6.4.1.2 Architecture

MiniBERT is identical to BERT other than in the following aspects:

- Instead of BERT's WordPiece tokenisation, MiniBERT uses the whole-word tokenisation described in Section 5.3.2.
- For simplicity, MiniBERT is trained only on BERT's masked language modelling task, not on next sentence prediction, and so does not use the [CLS] or [SEP] tokens. This has the consequence that MiniBERT cannot be fine-tuned in the way BERT can.
- MiniBERT has much smaller parameters than either $\textbf{BERT}_{\textbf{LARGE}}$ or $\textbf{BERT}_{\textbf{BASE}}$, as shown in Table 6.2.
- MiniBERT has a different training schedule, as described in Section 6.4.1.3.

	MiniBERT	**BERT$_{\text{BASE}}$**	**BERT$_{\text{LARGE}}$**
Embedding size d	256	768	1024
Number of attention heads	8	12	16
Number of Transformer layers	12	12	24
Maximum sequence length	128	512	512
Vocabulary size	86K	30K	30K
Number of training epochs	4	40	40
Total parameters	32M	110M	340M

Table 6.2: Parameters of MiniBERT, **BERT$_{\text{BASE}}$** and **BERT$_{\text{LARGE}}$**.

6.4.1.3 Training

MiniBERT was trained on batches consisting of 196 sequences of up to 128 tokens using the Adam optimiser (Kingma and Ba, 2014). The learning rate was increased linearly from 0 to 1e-4 over the first 10,000 batches and then left constant.

6.4.2 Clustering Parameters and Implementation Details

6.4.2.1 Dataset

In all experiments, D^+ consists of either 50,000 contexts containing w randomly selected from the corpus, or all such contexts if there are fewer than 50,000 in total. A random sample of 2,000,000 candidate negative contexts is taken from the corpus. When learning senses for w, D^- consists of the candidate negative contexts which do not contain w. This has the benefit that we do not need to take a new sample of negative contexts for each w.

6.4.2.2 Batching and Optimisation

Each batch contains 10,000 positive examples, and $K = 30$ negative examples per positive example. Substituting K for $\frac{D^-}{D^+}$ in the calculation of ρ, and rewriting $\frac{C^+}{C^-}$ as $\frac{p_w}{1-p_w}$, where p_w is the unigram probability of w in the corpus, we have

$$\rho = \frac{Kp_w}{1 - p_w}. \tag{6.25}$$

The objective $J(\Theta, \boldsymbol{\beta})$ is optimised using the Adam variant of stochastic gradient descent (Kingma and Ba, 2014). The model is trained for 1,000 batches, over

which the learning rate is decayed linearly from 1e-3 to 0. When using the distinctness loss term, r is increased linearly from 1 to its final value, which is 1.5 in these experiments. Gradually increasing the effect of the distinctness loss from nothing in this manner allows a diverse range of senses to be learned initially, and duplicate senses to then be eliminated.

6.4.2.3 Initialisation

The method is quite sensitive to the initialisation of the sense embeddings $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k$. Given that each element of MiniBERT's word embeddings is initialised to a value selected randomly from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \frac{1}{\sqrt{d}}$, it seems reasonable initialise the sense embeddings in the same manner. However it appears that better sense embeddings can be obtained for word w by setting the mean value of the initialising distribution for element i of the sense embeddings to be the ith element of MiniBERT's word embedding for w, i.e.

$$\boldsymbol{\theta}_{ji} \sim \text{Normal}\big(\mu = e_{wi}, \sigma = \frac{1}{\sqrt{d}}\big), \tag{6.26}$$

where $\boldsymbol{e}_w$ is MiniBERT's word embedding for word w. The bias terms $\boldsymbol{\beta}$ are initialised to 0.

6.4.3 Word Sense Induction

6.4.3.1 Method

Having obtained sense embeddings for word w, it is very simple to perform word sense induction (see Section 3.2.4) for w. The procedure for answering an instance from the SemEval-2010 and SemEval-2013 datasets where the focus word is w is as follows:

- Tokenise and lemmatise the instance to obtain a sequence of tokens $w_1, w_2, ..., w_n \in V$, where focus word $w_t = w$.
- Defining the context c to be the sequence $w_1, ..., w_{t-1}$, [MASK], $w_{t+1}, ..., w_n$, obtain the contextualised representation $\boldsymbol{y}(c)$ of the [MASK] token by run-

ning the sequence through the contextualiser (in this case MiniBERT) and taking the output representation at position t.

- Use $\boldsymbol{y}(c)$ and the sense embeddings $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_k$ learned for w to calculate the probabilities $\mathbb{P}(w_t \text{ has sense } w^{(i)} \mid c, w_t = w)$ for each $i \in \{1, 2, ..., k\}$ according to Equation 6.21.

- For SemEval-2010, simply label the instance with the sense $i \in \{1, 2, ..., k\}$ whose probability $\mathbb{P}(w_t \text{ has sense } w^{(i)} \mid c, w_t = w)$ is greatest.

- Recall that for SemEval-2013, an instance may be labelled with several senses, with a degree of applicability in the context specified for each. A threshold of $p_t = 0.2$ is chosen arbitrarily. Each instance is labelled with every sense $i \in \{1, 2, ..., k\}$ whose probability $\mathbb{P}(w_t \text{ has sense } w^{(i)} \mid c, w_t = w)$ is greater than p_t. The degree of applicability for each such sense is simply its probability $\mathbb{P}(w_t \text{ has sense } w^{(i)} \mid c, w_t = w)$.

6.4.3.2 Results

Table 6.3 shows the results of the MLCC method for learning sense embeddings on the SemEval-2010 and SemEval-2013 word sense induction tasks.

		SemEval-2010			**SemEval-2013**		
k	r	F-S	V-M	**AVG**	FBC	FNMI	**AVG**
5	1.5	67.8	34.6	48.4	66.6	16.1	32.8
8	1.0	55.6	32.5	42.5	59.4	15.0	29.8
8	1.5	66.6	34.4	47.9	66.5	15.7	32.2

Table 6.3: Performance of sense embeddings learned through MLCC on the SemEval-2010 and SemEval-2013 word sense induction datasets. The **AVG** score for each dataset is the geometric mean of the two sub-metrics. Results are shown for various settings of hyperparameters k (number of senses) and r (distinctness loss exponent).

Three different hyperparameter settings are tested. To determine the effectiveness of the distinctness loss, we compare the results when the distinctness loss exponent r is 1.5 to when it is 1. Note that $r = 1$ is equivalent to no distinctness loss, as J_d simplifies to 0 when $r = 1$. For $k = 8$, there is an improvement in all

metrics when distinctness loss is used, and a sizeable difference in the AVG scores for both datasets. To investigate the effect of the number of senses per word on performance, we compare the settings $k = 5$ and $k = 8$ with distinctness loss. The difference in performance between these settings is relatively small, suggesting that the distinctness loss is effective in eliminating duplicate senses, but that most words in the datasets have 5 or fewer senses, or that they have more than 5 senses but the method is not effective in learning the additional (likely rare) senses.

Table 6.4 compares the performance of the MLCC method to that of previous sense embedding methods on the SemEval-2010 and SemEval-2013 WSI tasks. The result shown is for the best parameter setting from Table 6.3, $k = 5$ and $r = 1.5$. MLCC comprehensively outperforms previous sense embedding methods on all metrics.

System	**Version**	**SemEval-2010**			**SemEval-2013**		
		F-S	V-M	**AVG**	FBC	FNMI	**AVG**
MLCC		**67.8**	**34.6**	**48.4**	**66.6**	**16.1**	**32.8**
Qiu et al. (2016)		-	-	-	56.9	6.7	19.5
SE-WSI-fix-cmp (Song et al., 2016)		54.3	16.3	29.8	-	-	-
AdaGram (Bartunov et al., 2016)		43.9	20.0	29.6	13.2	8.9	10.8
Arora et al. (2018)	$k = 5$	46.4	11.5	23.1	-	-	-

Table 6.4: Comparison of performance on SemEval-2010 and SemEval-2013 WSI tasks of the MLCC method for learning sense embeddings with $k = 5$ and $r = 1.5$ and that of four previous sense embedding methods. Note that SE-WSI-fix-cmp (Song et al., 2016) is based on the method of Neelakantan et al. (2014). **AVG** is the geometric mean of the two sub-metrics for each task.

Table 6.5 compares the performance of MLCC with that of two specialised WSI methods which have recently held the state-of-the-art on these datasets, those of Amplayo et al. (2019) and the slightly later Amrami and Goldberg (2019). Amrami and Goldberg's system uses the idea of *substitute vectors*, first devised by Başkaya et al. (2013). For each instance, a set of most likely words that could have occurred instead of the focus word is obtained from the output of a language model. These sets are then clustered, and each cluster is taken to correspond to a different sense of the focus word. Amrami and Goldberg report results using $\textbf{BERT}_{\textbf{LARGE}}$ as their language model. However, the $\textbf{BERT}_{\textbf{LARGE}}$ model contains more than 10 times as many parameters as MiniBERT, and was trained for 10 times longer (see Table

6.2). To provide a fair comparison, we evaluate the performance of Amrami and Goldberg's model with MiniBERT as the language model[1].

System	Version	SemEval-2010			SemEval-2013		
		F-S	V-M	**AVG**	FBC	FNMI	**AVG**
MLCC		67.8	34.6	48.4	**66.6**	16.1	32.8
Amrami and Goldberg (2019)	**BERT$_{\text{LARGE}}$**	**71.3**	**40.4**	**53.6**	64.0	**21.4**	**37.0**
	MiniBERT	67.5	29.2	44.4	61.0	13.0	28.1
AutoSense (Amplayo et al., 2019)		62.9	10.1	25.2	61.7	8.0	22.2

Table 6.5: Comparison of performance on SemEval-2010 and SemEval-2013 WSI tasks of the MLCC method for learning sense embeddings with $k = 5$ and $r = 1.5$ and that of two recent state-of-the-art models. **AVG** is the geometric mean of the two sub-metrics for each task. MLCC outperforms Amrami and Goldberg (2019)'s model when both methods use MiniBERT as their contextualiser/language model.

Amrami and Goldberg's method outperforms MLCC when **BERT$_{\text{LARGE}}$** is used. However when MiniBERT is used as the language model, MLCC is stronger on all metrics. This seems like a fairer comparison, since we have used MiniBERT as the contextualised model for MLCC.

[1]This was done by making appropriate modifications to the source code provided by the authors at `https://github.com/asafamr/bertwsi`.

Chapter 7

PolyLM

7.1 Motivation

While the MLCC method presented in Chapter 6 produced sense embeddings which perform well on word sense induction, the approach has two significant disadvantages:

1. There are several steps involved: training a contextualised word embedding model, extracting a number of negative and positive training examples for each word of interest, and performing the clustering. This is a complex and time-consuming process.

2. The word embeddings used by the contextualised word embedding model are still subject to the meaning conflation deficiency. Once this has occurred, it may become harder to separate out the meanings which have been conflated.

I now propose PolyLM, an end-to-end contextualised sense embedding model which addresses both of these disadvantages. PolyLM jointly learns sense embeddings and a contextualiser through training on masked language modelling. PolyLM can be easily applied to word-sense related NLP tasks with minimal or no additional training, addressing disadvantage 1 above. PolyLM uses only sense embeddings, not word embeddings, helping it to avoid disadvantage 2, meaning conflation.

7.2 Model

7.2.1 Overview

In a typical neural language model, each token w in the vocabulary V is assigned a single embedding $\boldsymbol{e}_w$, resulting in an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, where d is the embedding dimensionality. Some models, such as BERT (Devlin et al., 2019), also assign each word a bias value, resulting in a bias vector $\boldsymbol{b} \in \mathbb{R}^{|V|}$. In these models, the probability of w occurring in a context c is estimated as

$$\mathbb{P}(w \mid c) = \big[\text{softmax}\big(E\boldsymbol{y}(c) + \boldsymbol{b}\big)\big]_w, \tag{7.1}$$

where $\boldsymbol{y}(c) \in \mathbb{R}^d$ is a vector representation of c. Recall from Section 4.3.2 that in BERT, $\boldsymbol{y}(c)$ corresponds to the final output of multiple Transformer layers.

We wish to learn representations for individual senses, and so we assign an embedding to each sense in our sense inventory S, resulting in a matrix E with dimension $|S| \times d$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^{|S|}$. Note that this again assumes that we know the number of senses of each word *a priori*. Following Equation 7.1, we define the vector $\boldsymbol{p}(c) \in \mathbb{R}^{|S|}$ of sense probabilities in a context c as

$$\boldsymbol{p}(c) = \text{softmax}\big(E\boldsymbol{y}(c) + \boldsymbol{b}\big). \tag{7.2}$$

In other words, the estimated probability $\mathbb{P}(s \mid c)$ of sense s occurring in context c is given by

$$\mathbb{P}(s \mid c) = [\boldsymbol{p}(c)]_s = \frac{e^{\boldsymbol{e}_s^\top \boldsymbol{y}(c) + b_s}}{\sum\limits_{s' \in S} e^{\boldsymbol{e}_{s'}^\top \boldsymbol{y}(c) + b_{s'}}}. \tag{7.3}$$

Now recall Equation 1.1, the fundamental idea behind this thesis:

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \mathbb{P}(s \mid c).$$

Combining Equations 1.1 and 7.3, we have

$$\mathbb{P}(w \mid c) = \sum_{s \in S_w} \frac{e^{\boldsymbol{e}_s^\top \boldsymbol{y}(c) + b_s}}{\sum\limits_{s' \in S} e^{\boldsymbol{e}_{s'}^\top \boldsymbol{y}(c) + b_{s'}}}. \tag{7.4}$$

Having an expression for the probability of a word occurring in a given context allows us to formulate the problem of learning sense embeddings with a language modelling objective.

PolyLM is constructed from three components: the input layer, which represents the input tokens as aggregates of their sense embeddings, the disambiguation layer, which attempts to determine the contextually appropriate sense embeddings for the input, and the prediction layer, which implements the language modelling objective.

We adopt the masked language modelling (MLM) task used for training BERT (see Section 4.3.2.1). When training, we select a subset $T \subset \{1, 2, ..., n\}$ of the tokens in the input sequence as targets for prediction, and produce a masked version $c' = w'_1, w'_2, ..., w'_n$ of the original sequence $c = w_1, w_2, ..., w_n$ as follows: 15% of tokens are chosen at random as targets, of which 80% are replaced with a special [MASK] token, 10% are replaced with a random token, and 10% are left unchanged.

Figure 7.1 shows the architecture of the PolyLM model, each component of which will now be explained.

7.2.2 Input Layer

Recall from Section 4.1 the definition of a contextualiser as a function which maps a sequence of input representations $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathbb{R}^d$ to a corresponding sequence of output representations $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n \in \mathbb{R}^d$. Recurrent Neural Networks and Transformer architectures are both commonly used as contextualisers for language modelling. Typically the input representations are drawn from an embedding matrix $I \in \mathbb{R}^{|V| \times d}$. It has become common (e.g. BERT) to set I equal to E, the embedding matrix used at the language modelling output, as recommended by Press and Wolf (2017).

The issue of input representation poses a problem for a sense embedding model. The output embeddings $E \in \mathbb{R}^{|S| \times d}$ correspond to senses. It is not straightforward to tie the input and output embeddings as Press and Wolf suggest, because the senses of the words in an unannotated corpus are unknown, so it is not clear which of the sense embeddings for each word to use at the input. We solve this problem by

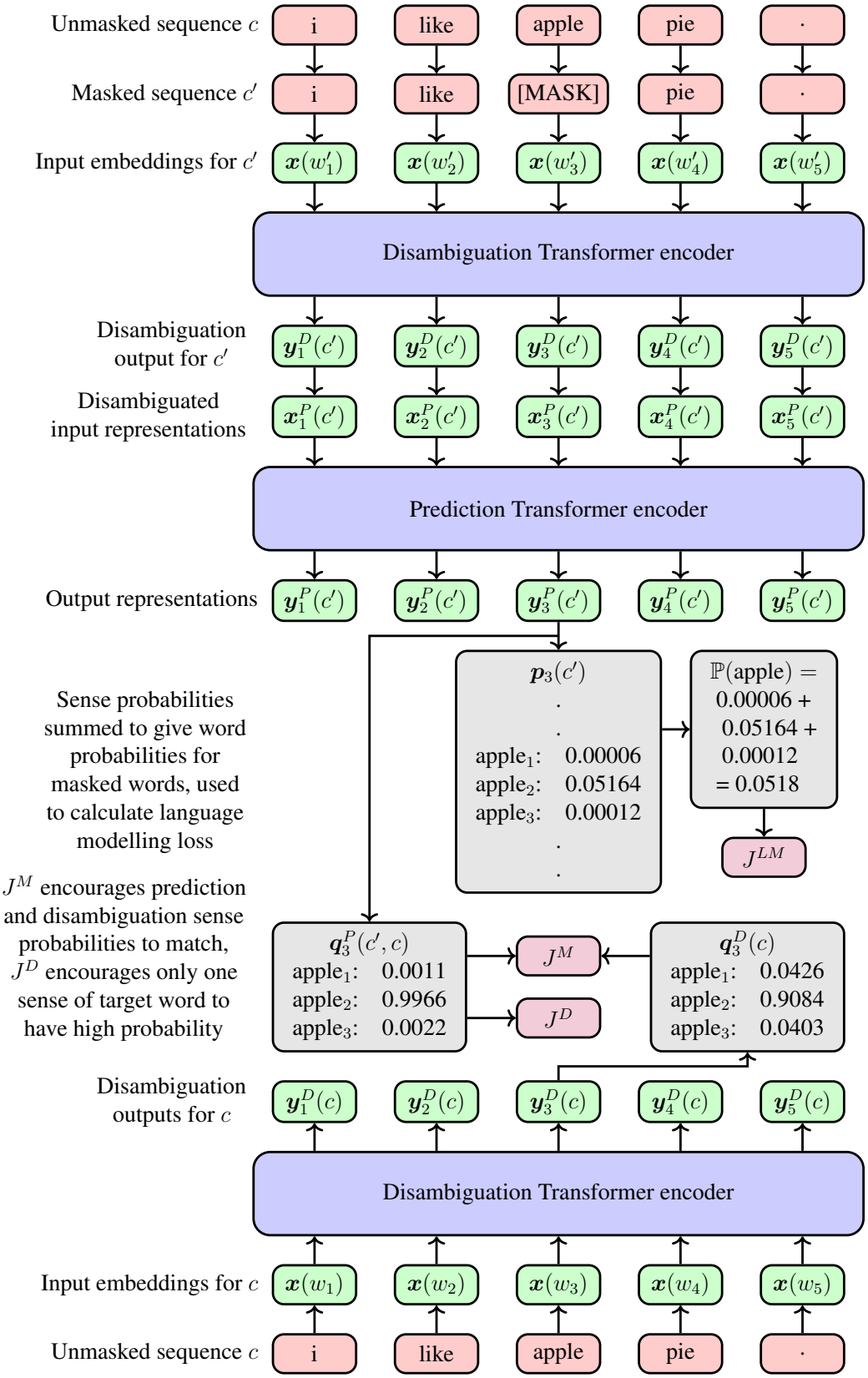


Figure 7.1: Architecture diagram for PolyLM, illustrated on the sentence "I like apple pie.", where the word "apple" is chosen as a target and masked.

setting the input representation of a word to be a convex combination of its sense embeddings, i.e.

$$\boldsymbol{x}(w) = \sum_{s \in S_w} \lambda_{ws} \boldsymbol{e}_s, \tag{7.5}$$

where $\boldsymbol{e}_s$ is the row of E corresponding to sense s, and $\boldsymbol{\lambda}_w$ is a learnable weight vector with the properties that $\sum_{s \in S_w} \lambda_{ws} = 1$ and $\boldsymbol{\lambda}_w \geq \mathbf{0}$ (in practice, $\boldsymbol{\lambda}_w$ is the softmax of an underlying, unconstrained variable vector).

7.2.3 Disambiguation Layer

The disambiguation layer attempts to infer the contextually appropriate sense embeddings for the input based on the conflated representations from the input layer.

Representations $\boldsymbol{x}(w'_1), \boldsymbol{x}(w'_2), ..., \boldsymbol{x}(w'_n)$ of c', calculated according to Equation 7.5, are fed into a contextualiser instance C^D, which outputs representations $\boldsymbol{y}^D_1(c'), \boldsymbol{y}^D_2(c'), ..., \boldsymbol{y}^D_n(c')$ (the superscript D denotes "Disambiguation layer"). These representations are used to calculate probability distributions $\boldsymbol{q}^D_1, \boldsymbol{q}^D_2, ..., \boldsymbol{q}^D_n$ over each sense of the tokens in the input:

$$\boldsymbol{q}^D_i(c') = \text{softmax}\big(E^{(w'_i)} \boldsymbol{y}^D_i(c') + \boldsymbol{b}^{(w'_i)}\big), \tag{7.6}$$

where $E^{(w'_i)}$ is a submatrix of E containing only the rows corresponding to senses of token w'_i, and similarly $\boldsymbol{b}^{(w'_i)}$ is a subvector of learnable bias vector $\boldsymbol{b} \in \mathbb{R}^{|S|}$. In other terms,

$$\mathbb{P}^D(w'_i \text{ has sense } s \mid c') = q^D_{is}(c') = \frac{e^{\boldsymbol{e}_s^\top \boldsymbol{y}^D_i(c') + b_s}}{\sum\limits_{s' \in S_{w'_i}} e^{\boldsymbol{e}_{s'}^\top \boldsymbol{y}^D_i(c') + b_{s'}}}, \tag{7.7}$$

where $s \in S_{w'_i}$.

The disambiguated representation of a token could simply be its highest-probability sense embedding in the context, but to allow gradients to flow through the disambiguation layer, it is instead defined to be the sum of the sense embeddings weighted by their probabilities:

$$\boldsymbol{x}^P_i(c') = \sum_{s \in S_{w'_i}} q^D_{is}(c') \boldsymbol{e}_s. \tag{7.8}$$

The disambiguated representation is written as $\boldsymbol{x}_i^P(c')$ because it is the input to the Prediction layer.

7.2.4 Prediction Layer

The prediction layer maps a sequence of disambiguated input representations onto a corresponding set of output representations, and from each output representation estimates the probability of every sense in the sense inventory occurring at the corresponding position of the sequence.

Disambiguated representations $\boldsymbol{x}_1^P(c'), \boldsymbol{x}_2^P(c'), ..., \boldsymbol{x}_n^P(c')$ are fed into another contextualiser instance C^P, which returns output representations $\boldsymbol{y}_1^P(c'), \boldsymbol{y}_2^P(c'), ..., \boldsymbol{y}_n^P(c')$. These are used to calculate a probability distribution over the entire sense inventory, as prescribed by Equation 7.2:

$$\boldsymbol{p}_i(c') = \text{softmax}(E\boldsymbol{y}_i^P(c') + \boldsymbol{b}). \tag{7.9}$$

We define an additional set of probabilities $\boldsymbol{q}^P$ analogous to $\boldsymbol{q}^D$ defined in Eq. 7.6:

$$\boldsymbol{q}_i^P(c', c) = \text{softmax}\big(E^{(w_i)}\boldsymbol{y}_i^P(c') + \boldsymbol{b}^{(w_i)}\big). \tag{7.10}$$

$\boldsymbol{q}_i^P$ takes both c' and the unmasked sequence c as arguments because we are interested in the sense probabilities of the words w_i that actually occurred. $\boldsymbol{q}_i^P$ will be used later for defining the loss function and in downstream tasks.

7.2.5 Loss Function

We seek to minimise a loss function J with three components, each of which is explained below:

$$\begin{aligned} J(c, c', T) = {} & J^{LM}(c, c', T) + \\ & J^D(c, c', T) + \\ & J^M(c, c', T) \end{aligned} \tag{7.11}$$

7.2.5.1 Language Modelling Loss

The language modelling loss J^{LM} is defined as the mean negative log likelihood of the target tokens occurring, as in a typical language model:

$$J^{LM}(c, c', T) = \frac{-1}{|T|} \sum_{i \in T} \log \mathbb{P}(\text{token } i \text{ is } w_i \mid c') \tag{7.12}$$

$$= \frac{-1}{|T|} \sum_{i \in T} \log \sum_{s \in S_{w_i}} \mathbb{P}(\text{token } i \text{ is } w_i \text{ with sense } s \mid c') \tag{7.13}$$

$$= \frac{-1}{|T|} \sum_{i \in T} \log \sum_{s \in S_{w_i}} p_{is}(c'), \tag{7.14}$$

where $\boldsymbol{p}_i$ is as defined in Eq. 7.9.

7.2.5.2 Distinctness Loss

Recall that we assume in advance a number of senses for each word. In practice we guess a relatively high number to avoid missing senses. When we overestimate the number of senses, we find that two or more different sense embeddings for a word converge to essentially the same meaning. As for maximum likelihood clustering, we employ a "distinctness loss" term whose aim is to ensure that each sense has a distinct meaning by "killing off" unnecessary senses, i.e. causing them to have very low probability in all contexts.

Recall from Section 6.3 the assumption underlying the formulation of the distinctness loss: that if the senses corresponding to a word w are distinct, then in contexts where w occurs, we would expect one of these senses to have a high estimated probability of occurring, and the rest to have a low probability. We observed that the function

$$f(\boldsymbol{p}) = \sum_i p_i^r, \tag{7.15}$$

where $r > 1$ is a constant and $\boldsymbol{p}$ is a probability distribution, has the property that it is maximised when one of its elements is 1 and the rest are 0, and is generally largest when its values are most unevenly distributed. This suggests a distinctness loss of the form

$$J^D(c, c', T) = \frac{-\lambda}{|T|} \sum_{i \in T} \sum_{s \in S_{w_i}} \left(q_{is}^P(c', c)\right)^r, \tag{7.16}$$

where λ and r are both hyperparameters; the negation is present because the loss function is formulated with a minimisation objective. However the form of the distinctness function used involves a logarithmic transformation, and sets λ to $\frac{1}{r}$:

$$J^D(c, c', T) = \frac{-1}{r|T|} \sum_{i \in T} \log \sum_{s \in S_{w_i}} \big(q^P_{is}(c', c)\big)^r. \tag{7.17}$$

The justification for this formulation is that it causes the gradients of the loss function to have an intuitively appealing property, which I will now explain.

Consider the derivative of the language modelling loss for one particular target position $i \in T$ with respect to the pre-softmax scores $\boldsymbol{e}_k^\top \boldsymbol{y}_i^P + b_k$ of the target word w_i's sense $k \in S_{w_i}$. For brevity, we will define $y_k = \boldsymbol{e}_k^\top \boldsymbol{y}_i^P + b_k$.

$$\begin{aligned}
-\frac{\partial}{\partial y_k} J^{LM}(c, c', \{i\}) &= \frac{\partial}{\partial y_k} \log \sum_{s \in S_{w_i}} \big[\text{softmax}(E\boldsymbol{y}_i^P + \boldsymbol{b})\big]_s \\
&= \frac{\partial}{\partial y_k} \log \sum_{s \in S_{w_i}} \frac{e^{\boldsymbol{e}_s^\top \boldsymbol{y}_i^P + b_s}}{\sum_{s' \in S} e^{\boldsymbol{e}_{s'}^\top \boldsymbol{y}_i^P + b_{s'}}} \\
&= \frac{\partial}{\partial y_k} \log \sum_{s \in S_{w_i}} \frac{e^{y_s}}{\sum_{s' \in S} e^{y_{s'}}} \\
&= \frac{\partial}{\partial y_k} \log \frac{\sum_{s \in S_{w_i}} e^{y_s}}{\sum_{s \in S_{w_i}} e^{y_s} + \sum_{s \in S \setminus S_{w_i}} e^{y_s}} \\
&= \frac{\partial}{\partial y_k} \log \frac{\sum_{s \in S_{w_i}} e^{y_s}}{\sum_{s \in S_{w_i}} e^{y_s} + C}, \text{ where } C = \sum_{s \in S \setminus S_{w_i}} e^{y_s}, \\
&= \frac{\partial}{\partial y_k} \Bigg(\log \sum_{s \in S_{w_i}} e^{y_s} - \log \Big(\sum_{s \in S_{w_i}} e^{y_s} + C \Big) \Bigg) \\
&= \frac{\frac{\partial}{\partial y_k} \sum_{s \in S_{w_i}} e^{y_s}}{\sum_{s \in S_{w_i}} e^{y_s}} - \frac{\frac{\partial}{\partial y_k} (\sum_{s \in S_{w_i}} e^{y_s} + C)}{\sum_{s \in S_{w_i}} e^{y_s} + C} \\
&= \frac{e^{y_k}}{\sum_{s \in S_{w_i}} e^{y_s}} - \frac{e^{y_k}}{\sum_{s \in S_{w_i}} e^{y_s} + C} \\
&= \frac{e^{y_k}}{\sum_{s \in S_{w_i}} e^{y_s}} - \frac{e^{y_k}}{\sum_{s \in S} e^{y_s}} \\
&= q^P_{ik}(c', c) - p_{ik}(c).
\end{aligned} \tag{7.18}$$

Since $q^P_{ik} > p_{ik}$, $\frac{\partial}{\partial y_k} J^{LM}(c, c', \{i\})$ will always be negative, meaning that every sense embedding for the target word will always move towards the contextualised representation $\boldsymbol{y}_i^P$. This is undesirable, because it means that even senses which are irrelevant in a context will receive a positive update.

Now consider the derivatives of the distinctness loss:

$$
\begin{aligned}
-\frac{\partial}{\partial y_k} J^D(c, c', \{i\}) &= \frac{\partial}{\partial y_k} \frac{1}{r} \log \sum_{s \in S_{w_i}} \left(q^P_{is}(c', c)\right)^r \\
&= \frac{1}{r} \frac{\partial}{\partial y_k} \log \sum_{s \in S_{w_i}} \left(\frac{e^{y_s}}{\sum_{s' \in S_{w_i}} e^{y_{s'}}} \right)^r \\
&= \frac{1}{r} \frac{\partial}{\partial y_k} \log \sum_{s \in S_{w_i}} \frac{e^{r y_s}}{(\sum_{s' \in S_{w_i}} e^{y_{s'}})^r} \\
&= \frac{1}{r} \frac{\partial}{\partial y_k} \log \frac{\sum_{s \in S_{w_i}} e^{r y_s}}{(\sum_{s \in S_{w_i}} e^{y_s})^r} \\
&= \frac{1}{r} \frac{\partial}{\partial y_k} \left(\log \sum_{s \in S_{w_i}} e^{r y_s} - \log \left(\sum_{s \in S_{w_i}} e^{y_s} \right)^r \right) \\
&= \frac{1}{r} \left(\frac{\frac{\partial}{\partial y_k} \sum_{s \in S_{w_i}} e^{r y_s}}{\sum_{s \in S_{w_i}} e^{r y_s}} - \frac{r \frac{\partial}{\partial y_k} \sum_{s \in S_{w_i}} e^{y_s}}{\sum_{s \in S_{w_i}} e^{y_s}} \right) \\
&= \frac{1}{r} \left(\frac{r e^{r y_k}}{\sum_{s \in S_{w_i}} e^{r y_s}} - \frac{r e^{y_k}}{\sum_{s \in S_{w_i}} e^{y_s}} \right) \\
&= \frac{e^{r y_k}}{\sum_{s \in S_{w_i}} e^{r y_s}} - q^P_{ik}(c', c). \qquad (7.19)
\end{aligned}
$$

When $r > 1$, $\frac{e^{r y_k}}{\sum_{s \in S_{w_i}} e^{r y_s}}$ is a "sharpened" version of $q^P_{ik}(c', c)$: it is larger than q^P_{ik} when q^P_{ik} is large, and smaller when q^P_{ik} is small.

Now consider the derivative of the sum of the language modelling and distinctness losses:

$$
\begin{aligned}
-\frac{\partial}{\partial y_k} \left(J^{LM} + J^D(c, c', \{i\}) \right) &= -\frac{\partial}{\partial y_k} J^{LM}(c, c', \{i\}) - \frac{\partial}{\partial y_k} J^D(c, c', \{i\}) \\
&= q^P_{ik}(c', c) - p_{ik}(c) + \frac{e^{r y_k}}{\sum_{s \in S_{w_i}} e^{r y_s}} - q^P_{ik}(c', c) \\
&= \frac{e^{r y_k}}{\sum_{s \in S_{w_i}} e^{r y_s}} - p_{ik}(c) \\
&= q^{\text{sharp}}_{ik}(c', c) - p_{ik}(c). \qquad (7.20)
\end{aligned}
$$

Comparing Equations 7.18 and 7.20, we see that the addition of the distinctness loss results in even stronger reinforcement for senses which are highly applicable in the context, and even weaker (possibly negative) reinforcement for senses which are inapplicable. This encourages only one sense of a word to have high probability in a given context, as desired.

7.2.5.3 Match Loss

Without extra supervision, the disambiguation layer tends to very quickly allocate almost all of the probability mass for a word to a single one of its senses. This appears to be due to a "rich get richer" effect in Equation 7.8, where the sense embedding with the highest weight has larger gradients associated with it.

A more reliable source of sense probabilities is the output of the prediction layer, as this is more closely associated with the ground truth. Therefore the disambiguation sense probabilities $\boldsymbol{q}^D$ are encouraged to be similar to the prediction sense probabilities $\boldsymbol{q}^P$ by adding a sense probability "match loss," which is proportional to the cosine similarity between $\boldsymbol{q}^D$ and $\boldsymbol{q}^P$.

Because $\boldsymbol{q}_i^D(c')$ is meaningless when token i is replaced with [MASK], when calculating the match loss, the disambiguation layer is evaluated on the unmasked sequence, obtaining $\boldsymbol{q}_i^D(c)$. The match loss is defined as

$$J^M(c, c', T) = \frac{-\lambda^M}{|T|} \sum_{i \in T} \frac{\boldsymbol{q}_i^D \cdot \boldsymbol{q}_i^P}{\|\boldsymbol{q}_i^D\| \|\boldsymbol{q}_i^P\|}, \tag{7.21}$$

where $\boldsymbol{q}_i^D$ and $\boldsymbol{q}_i^P$ are shorthand for $\boldsymbol{q}_i^D(c)$ and $\boldsymbol{q}_i^P(c', c)$ respectively, and λ^M is a hyperparameter.

As we wish the disambiguation layer to learn from the prediction layer rather than the other way around, we do not allow gradients from the match loss to propagate through $\boldsymbol{q}_i^P$.

7.2.6 Implementation Details and Parameters

7.2.6.1 Corpus and Preprocessing

PolyLM is trained on the same corpus with the same preprocessing and lemmatisation used to train MiniBERT (see Sections 5.3.1, 5.3.2 and 6.4.1.1).

7.2.6.2 Contextualisers

One of the advantages of PolyLM is that it can be used with any type of contextualiser. For experiments, both the disambiguation and prediction contextualisers C^D

	PolyLM	MiniBERT	**BERT$_{\text{BASE}}$**	**BERT$_{\text{LARGE}}$**
Embedding size d	128	256	768	1024
Number of attention heads	8	8	12	16
Number of Transformer layers	4 (C^D), 8 (C^P)	12	12	24
Maximum sequence length	128	128	512	512
Vocabulary size	86K	86K	30K	30K
Number of training epochs	4	4	40	40
Total parameters	30M	32M	110M	340M

Table 7.1: Parameters of PolyLM, MiniBERT, **BERT$_{\text{BASE}}$** and **BERT$_{\text{LARGE}}$**.

and C^P, like MiniBERT, use BERT's implementation of the Transformer encoder architecture.

7.2.6.3 Parameters

Due to the prohibitive computational cost of training a model of **BERT$_{\text{LARGE}}$**'s size, PolyLM has significantly smaller dimensions, as shown in Table 7.1. To keep the total number of embeddings reasonable, only the ~10,000 tokens which occur more than 20,000 times in the training corpus, or appear as focuses in the evaluation datasets, are allowed to have multiple senses. Specifically, these tokens are assigned a fixed number of k embeddings, and other tokens a single embedding.

7.2.7 Training

PolyLM was trained for 4 epochs with Adam (Kingma and Ba, 2014), using batches consisting of 64 sequences of length 128. The learning rate was increased linearly from 0 to 1e-4 over the first 10,000 batches and then left constant. The hyperparameters λ^M and r specific to PolyLM's loss function were also increased linearly during training, λ^M from 0 to 0.1 over the first 2 epochs, and r from 1.0 to 1.5 over all 4 epochs.

7.3 Experiments

As for MLCC, PolyLM is evaluated primarily on word sense induction (WSI), as WSI has been a standard evaluation task for previous sense embedding models.

To demonstrate the flexibility of PolyLM, particularly as a result of the fact that it produces a probability distribution over word senses, we show that it can also be easily used to obtain good results on two tasks to which unsupervised sense embeddings have not previously been applied, word sense disambiguation (WSD) and word-in-context (WiC) (Pilehvar and Camacho-Collados, 2019).

To determine the value added by the various components of the model, several versions of PolyLM are trained:

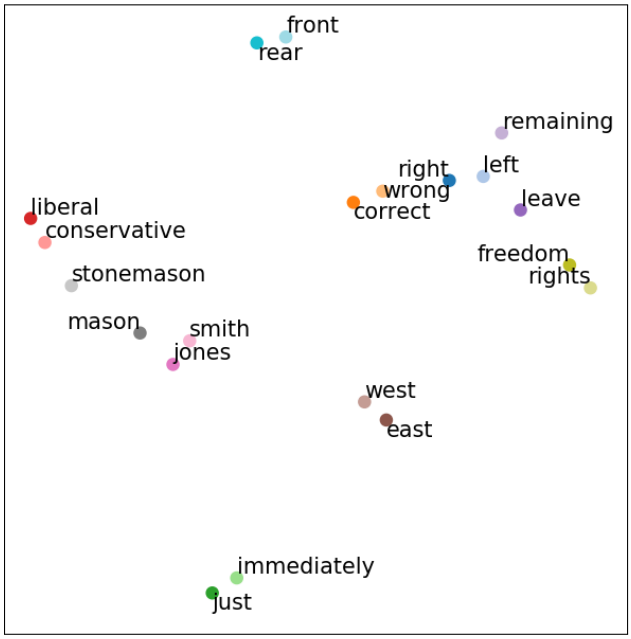
- The full model, as described above, with $k = 5$.
- The full model with $k = 8$.
- The full model with $k = 12$.
- A version trained on the unlemmatised corpus, $k = 12$. At evaluation time, focus words are replaced with their lemmatised form.
- A version without distinctness loss (equivalent to $r = 1$), $k = 12$.
- A version without a disambiguation layer, i.e. the output of the input layer is fed directly into the prediction layer, and there is no match loss. $k = 12$.

PolyLM produces two sets of word sense probabilities, $\boldsymbol{q}^D$ from the disambiguation layer, and $\boldsymbol{q}^P$ from the prediction layer. Either can be used for downstream tasks. Generally the probabilities $\boldsymbol{q}^P$ yield the best performance on the evaluation tasks, but to demonstrate that the disambiguation layer alone is effective we also show results using $\boldsymbol{q}^D$. From now on, $\boldsymbol{q}$ will be used in contexts where both $\boldsymbol{q}^P$ and $\boldsymbol{q}^D$ are applicable. Except where an experiment is explicitly labelled with $\boldsymbol{q}^D$, $\boldsymbol{q}^P$ is used.

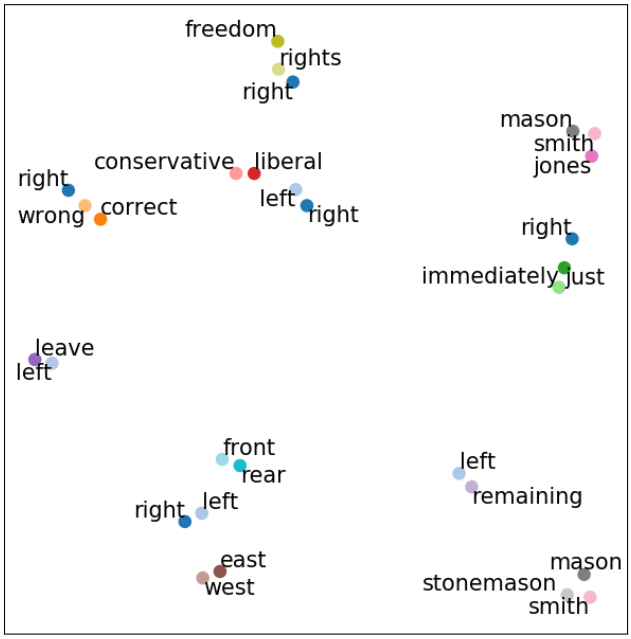
Figure 7.2 illustrates the reduction in meaning conflation achieved through PolyLM's use of sense embeddings versus MiniBERT's word embeddings.

7.3.1 Word Sense Induction

PolyLM can be used for WSI without any further training. The method used is similar to that used for applying sense embeddings learned through MLCC to WSI



(a) Word embeddings (MiniBERT)



(b) Sense embeddings (PolyLM)

Figure 7.2: An illustration of the meaning conflation deficiency, showing selected word and sense embeddings learned by MiniBERT and PolyLM visualised using t-SNE (Maaten and Hinton, 2008) and adjustText (Flyamer, 2017). The occurrence of closely related polysemous words nearby in the word embedding space (i.e. *left* and *right*) causes unrelated words to be closer together (e.g. *left* and *wrong*) and related words to be further apart (e.g. *right* and *east*) than they otherwise would be. The use of sense embeddings avoids such distortion. PolyLM is capable of detecting comparatively rare word senses, such as the political senses of *left* and *right*, and the use of *smith* and *mason* to refer to tradespeople. Note that the models used to create these data were trained on unlemmatised corpora.

(see Section 6.4.3.1): for the SemEval-2010 dataset, each instance c is labelled with the sense of the focus word w_t which has the highest predicted probability, i.e. $\text{argmax}_{s \in S_{w_t}} \boldsymbol{q}_{ts}(c', c)$. c' is formed from c by replacing w_t with [MASK]. For SemEval-2013, we consider a sense applicable if it has a predicted probability $q_{ts}(c', c) > p_{\text{thresh}}$, and the weight assigned to each applicable sense is its probability $q_{ts}(c', c)$. As in Section 6.4.3.1, p_{thresh} is set to 0.2.

Results for PolyLM with the various settings detailed above are shown in Table 7.2. PolyLM performs slightly better with a smaller rather than larger k. Given that we expect most of the focus words in the datasets to have fewer than 12 senses, this suggests that the distinctness loss is fairly although not entirely effective at eliminating duplicate senses. Comparing the $k = 12, \boldsymbol{q}^P$ version with the various ablated versions, we see that the performance is somewhat better when the sense probabilities $\boldsymbol{q}^P$ from the prediction layer are used versus when the sense probabilities $\boldsymbol{q}^D$ from the disambiguation layer are used. A similar difference is observed when lemmatisation is used compared to when the corpus is not lemmatised. The results show that using the distinctness loss is responsible for a large gain in performance. When no disambiguation layer is used, the performance is better on the F-S and FBC metrics and worse on the V-M and FNMI metrics, and slightly worse overall. Recall from Section 3.2.4 that the former two metrics tend to favour systems which learn a fewer number of senses per word, while the latter two favour those which learn a larger number. It appears that when there is no disambiguation layer, PolyLM is less likely to learn multiple embeddings corresponding to the same meaning, increasing the F-S and FBC metrics, but is also less likely to detect rare word senses, decreasing the V-M and FNMI metrics.

Table 7.3 compares the performance of PolyLM, maximum likelihood contextual clustering (MLCC) and previous sense embedding methods on WSI. PolyLM performs slightly worse than MLCC on the SemEval-2010 dataset, somewhat better on the SemEval-2013 dataset, and much better than previous methods on both datasets. The fact that PolyLM performs better than MLCC on the V-M and FNMI metrics but worse on the F-S and FBC metrics suggests that it learns a more diverse

Settings	SemEval-2010			SemEval-2013		
	F-S	V-M	**AVG**	FBC	FNMI	**AVG**
$k = 5, \boldsymbol{q}^P$	**64.0**	36.3	**48.2**	**63.9**	20.1	**35.8**
$k = 8, \boldsymbol{q}^P$	60.9	37.8	47.9	62.2	19.8	35.1
$k = 12, \boldsymbol{q}^P$	56.9	**38.1**	46.6	60.1	**20.2**	34.9
$k = 12, \boldsymbol{q}^D$	55.2	33.7	43.1	59.9	16.2	31.2
$k = 12, \boldsymbol{q}^P$, unlemmatized	56.2	34.0	43.7	57.8	16.8	31.1
$k = 12, \boldsymbol{q}^P$, no distinctness loss	46.0	34.5	39.8	51.8	15.0	27.9
$k = 12, \boldsymbol{q}^P$, no disambiguation layer	61.6	35.0	46.4	62.6	19.2	34.6

Table 7.2: Comparison of the performance of PolyLM with various parameter settings and ablations on the SemEval-2010 and SemEval-2013 word sense induction tasks. **AVG** is the geometric mean of the two sub-metrics for each task.

range of word senses, but also has a greater tendency to learn duplicate senses.

System	Version	SemEval-2010			SemEval-2013		
		F-S	V-M	**AVG**	FBC	FNMI	**AVG**
PolyLM	$k = 5$	64.0	**36.3**	48.2	63.9	**20.1**	**35.8**
MLCC	$k = 5$	**67.8**	34.6	**48.4**	**66.6**	16.1	32.8
(Qiu et al., 2016)		-	-	-	56.9	6.7	19.5
SE-WSI-fix-cmp (Song et al., 2016)		54.3	16.3	29.8	-	-	-
AdaGram (Bartunov et al., 2016)		43.9	20.0	29.6	13.2	8.9	10.8
Arora et al. (2018)	$k = 5$	46.4	11.5	23.1	-	-	-

Table 7.3: Comparison of the performance of PolyLM and MLCC on the SemEval-2010 and SemEval-2013 word sense induction tasks with that of four previous sense embedding methods. **AVG** is the geometric mean of the two sub-metrics for each task.

Recall from Section 6.4.3.2 that when MiniBERT is used as the contextualised model for both Amrami and Goldberg (2019)'s state-of-the-art WSI method and for the MLCC method, MLCC performs better. Since PolyLM is also a language model, it can like MiniBERT be used as the "engine" for Amrami and Goldberg's method; the results of doing so are included in Table 7.4. Interestingly, Amrami and Goldberg + PolyLM outperforms Amrami and Goldberg + MiniBERT by a large margin, despite the fact that PolyLM and MiniBERT are similarly sized models. This suggests that PolyLM is more sensitive to sense distinctions than a single-sense model such as BERT. This is in keeping with our hypothesis that the meaning conflation deficiency hampers word embedding models' understanding of polysemy.

System	Version	SemEval-2010			SemEval-2013		
		F-S	V-M	**AVG**	FBC	FNMI	**AVG**
PolyLM	$k = 5$	64.0	**36.3**	48.2	63.9	**20.1**	**35.8**
MLCC	$k = 5$	**67.8**	34.6	48.4	**66.6**	16.1	32.8
Amrami and Goldberg (2019)	MiniBERT	67.5	29.2	44.4	61.0	13.0	28.1
	PolyLM	66.3	36.2	**49.0**	60.8	19.9	34.7

Table 7.4: Comparison of performance on SemEval-2010 and SemEval-2013 WSI tasks of PolyLM and MLCC with that of Amrami and Goldberg (2019)'s recent state-of-the-art model when MiniBERT and PolyLM ($k = 12$) are used as its language model. **AVG** is the geometric mean of the two sub-metrics for each task.

7.3.2 Word-in-Context

Word-in-Context, or WiC (Pilehvar and Camacho-Collados, 2019), is a binary classification task in which each instance consists of two sentences containing an occurrence of a given focus word. An instance is labelled as positive if the focus word has the same sense in both sentences, and false otherwise.

A strong baseline for WiC is given by classifying examples according to whether the cosine similarity of $\textbf{BERT}_{\textbf{LARGE}}$'s contextualised representations for the focus word in the two sentences exceeds a threshold learned on the training set. Wang et al. (2019) showed that fine-tuning BERT could produce better results than using its contextualised representations directly. The most successful techniques (Levine et al., 2019; Peters et al., 2019) have involved augmenting BERT with supervision from knowledge bases such as WordNet.

Using PolyLM, we can calculate an estimate for the probability that the focus word w has the same sense in sentences c_1 and c_2 as follows:

$$\mathbb{P}(w \text{ has same sense in } c_1 \text{ and } c_2) = \sum_{s \in S_w} \mathbb{P}(w \text{ has sense } s \text{ in } c_1 \text{ and } c_2) \tag{7.22}$$

$$= \sum_{s \in S_w} q_{is}(c'_1, c_1)\, q_{js}(c'_2, c_2) \tag{7.23}$$

$$= \boldsymbol{q}_i(c'_1, c_1)^\top \boldsymbol{q}_j(c'_2, c_2), \tag{7.24}$$

where i and j are the focus positions in the two sentences, and c'_1 and c'_2 are again created by masking only the focus positions. We call $\boldsymbol{q}_i(c'_1, c_1)^\top \boldsymbol{q}_j(c'_2, c_2)$ the "probability score."

We evaluate PolyLM in 3 ways: using cosine similarity of contextualised rep-

resentations as described above (CS), using probability score (PS) with a learned threshold, or using logistic regression with both CS and PS as features (LR).

Results are shown in Table 7.5, and a comparison with other systems in Table 7.6. PolyLM outperforms MiniBERT and even **BERT$_{\text{LARGE}}$** without fine-tuning. PolyLM as described in this thesis cannot be fine-tuned on WiC because we do not use BERT's sentence ordering task for training.

Version	**Train + Dev. set**			**Test set**
	CS	PS	LR	
$k = 8, \boldsymbol{q}^P$	66.9	64.6	68.7	66.0
$k = 12, \boldsymbol{q}^P$	67.1	65.6	69.1	**66.7**
$k = 12, \boldsymbol{q}^D$	-	65.3	-	62.9
$k = 12, \boldsymbol{q}^P$, unlemmatized	65.5	63.7	66.7	65.3
$k = 12, \boldsymbol{q}^P$, no distinctness loss	67.6	65.9	69.2	66.5
$k = 12, \boldsymbol{q}^P$, no disambiguation layer	66.7	61.2	68.1	65.5

Table 7.5: Accuracy of PolyLM with various parameter settings and ablations on the Word-in-Context task. Note that due to the limited number of submissions allowed to the WiC test set, only the LR method is evaluated on the test set (except $k = 12, \boldsymbol{q}^D$, which is evaluated on PS, since output contextualised representations are unavailable when only the disambiguation layer is used).

System	**Test set**
PolyLM	66.7
SenseBERT (Levine et al., 2019)	**72.1**
KnowBERT (Peters et al., 2019)	70.9
BERT$_{\text{LARGE}}$ (Wang et al., 2019)	69.5
BERT$_{\text{LARGE}}$ (Pilehvar and Camacho-Collados, 2019)	65.5
MiniBERT	63.9

Table 7.6: Accuracy of PolyLM and various other systems on the Word-in-Context task. MiniBERT is evaluated using CS.

7.3.3 Word Sense Disambiguation

We adopt the word sense disambiguation (WSD) framework of (Raganato et al., 2017a), which contains five datasets standardised to the WordNet 3.0 inventory, as described in Section 3.2.3. Before we can use PolyLM for WSD, we must map from its induced sense inventory to WordNet's.

Given $\mathbb{P}(\text{word } w \text{ has sense } s \text{ in context } c)$ for all $s \in S_w$, we want to be able to calculate $\mathbb{P}(\text{word } w \text{ has WordNet sense } s' \text{ in context } c)$ for all $s' \in S'_w$, where S'_w is WordNet's set of senses for w. To this end, we assume the existence of a mapping matrix $M^w \in \mathbb{R}^{|S_w| \times |S'_w|}$, where $M^w_{ss'}$ gives the probability that in a randomly selected context containing w with sense $s \in S_w$, the appropriate WordNet sense for w is $s' \in S'_w$. We can write $M^w_{ss'}$ as

$$M^w_{ss'} = \mathbb{P}(Q_w = s' \mid P_w = s) \tag{7.25}$$

$$= \frac{\mathbb{P}(P_w = s \mid Q_w = s')\, \mathbb{P}(Q_w = s')}{\mathbb{P}(P_w = s)}, \tag{7.26}$$

where P_w and Q_w are random variables representing the PolyLM and the WordNet sense respectively of instances of w.

We require some labelled training data to obtain an estimate of $\mathbb{P}(P_w = s \mid Q_w = s')$. Suppose we have m training examples $c_1, c_2, ..., c_m$ in which w appears with WordNet sense s' in positions $t_1, t_2, ..., t_m$ respectively. Then we calculate

$$\mathbb{P}(P_w = s \mid Q_w = s') = \frac{1}{m} \sum_{j=1}^{m} q_{t_j s}(c'_j, c_j) \tag{7.27}$$

We calculate $\mathbb{P}(Q_w = s')$ according to the proportion of training examples for word w which have WordNet sense s', and $\mathbb{P}(P_w = s)$ by taking the mean predicted probability of PolyLM sense s across all training examples for word w.

When labelling a test instance c with focus word w at position t, we have

$$\mathbb{P}(Q_w = s' \mid c) = \sum_{s \in S_w} \mathbb{P}(Q_w = s' \mid P_w = s)\, \mathbb{P}(P_w = s \mid c) \tag{7.28}$$

$$= \sum_{s \in S_w} M^w_{ss'}\, q_{ts}(c', c). \tag{7.29}$$

Thus the vector of WordNet sense probabilities $\boldsymbol{p}^{WN}_t(c) \in \mathbb{R}^{|S'_w|}$ is given by

$$\boldsymbol{p}^{WN}_t(c) = \boldsymbol{q}_t(c', c)^\top M^w. \tag{7.30}$$

Focus word w at index t in context c is labelled with the most probable WordNet sense, i.e. $\text{argmax}_{s' \in S'_w}\, p^{WN}_{ts'}(c)$.

The standard SemCor dataset (Miller et al., 1994) is used to find mapping matrices M^w for each focus word w in the test set, and it is augmented with the usage

examples and glosses contained in WordNet for better sense coverage. Specifically, for a sense of the word t with gloss G, an example c is created using the pattern *"t" means "G"*.

WordNet contains a number of compound lemmas consisting of multiple words such as "rural_area." PolyLM cannot handle these types of lemmas, so first-sense backoff is applied for these. That is, they are always labelled as belonging to the first WordNet sense, which is a standard technique in WSD for dealing with unknown lemmas.

Results of PolyLM on WSD are shown in Table 7.7. Unlike for WSI, PolyLM without distinctness loss performs strongly. This is because multiple PolyLM word senses can effectively be mapped to the same WordNet sense, so having duplicate senses does not harm performance.

Version	**All**
$k = 8, \boldsymbol{q}^P$	72.6
$k = 12, \boldsymbol{q}^P$	73.4
$k = 12, \boldsymbol{q}^D$	72.7
$k = 12, \boldsymbol{q}^P$, no distinctness loss	**73.5**
$k = 12, \boldsymbol{q}^P$, no disambiguation layer	71.7

Table 7.7: Accuracy of PolyLM with various parameter settings and ablations on the WSD framework of Raganato et al. (2017a).

System	**All**	**All except SemEval-2007**
PolyLM	73.4	73.9
GlossBERT (Huang et al., 2019)†	-	**77.0**
Vial et al. (2019)†	**75.6**	-
LMMS (Loureiro and Jorge, 2019)†	75.4	-
HCAN (Luo et al., 2018)	-	71.1

Table 7.8: Accuracy of PolyLM and other recent WSD methods on the framework of Raganato et al. (2017a). Some systems use the Semeval-2007 dataset as a development set, and report test results on the other datasets. † - uses pre-trained BERT.

Table 7.8 compares PolyLM's WSD performance with that of several other recent methods. PolyLM outperforms previous methods except those which use pre-trained BERT, despite no particular effort to optimise its performance on WSD, such as by using lexical knowledge like Vial et al. (2019) and Loureiro and Jorge (2019).

GlossBERT (Huang et al., 2019) uses BERT fine-tuning, which as discussed previously is not possible with PolyLM without training on a sentence ordering task.

7.4 Summary

I have proposed PolyLM, an end-to-end, unsupervised, contextualised sense embedding model which reduces meaning conflation by using only sense embeddings, not word embeddings, and a "disambiguation layer" which determines the contextually appropriate sense embedding for each word at the input. PolyLM performs much better than previous sense embedding models on word sense induction. It also outperforms Amrami and Goldberg (2019)'s state-of-the-art specialised WSI method when the comparably-sized MiniBERT is used as its language model. This does not prove however that if PolyLM were scaled up to $\textbf{BERT}_{\textbf{LARGE}}$'s size, it would outperform Amrami and Goldberg's method with $\textbf{BERT}_{\textbf{LARGE}}$ as its language model. Unlike previous work on unsupervised sense embeddings, I showed that PolyLM can easily be applied to word sense disambiguation and the word-in-context task, with results that seem reasonable despite no particular effort being made to optimise the performance using lexical knowledge, for instance.

Chapter 8

Conclusions

The aim of this research was to advance the study of sense embeddings by investigating how powerful recent contextualisation techniques can be exploited to learn better unsupervised sense embeddings. Contextualisers learn transferable knowledge about language through training on the unsupervised task known as "language modelling." The aim of language modelling is to produce the best possible estimates of the probabilities of various words occurring in a given context. To bridge the gap between predicting the occurrence of words and the occurrence of word *senses*, I proposed a simple hypothesis which states that the probability of a word occurring in a context is equal to the sum of the probabilities of its individual senses occurring. I showed that this hypothesis could be used to develop a simple model based on Mikolov et al. (2013a)'s Skip-gram model which learns reasonable sense embeddings.

I proposed a method called "maximum likelihood contextual clustering" (MLCC), based on the same hypothesis, for learning sense embeddings aligned with the embedding space of a given pre-trained contextualised word embedding model. The resulting embeddings were evaluated on two standard word sense induction datasets, on which they outperformed previous sense emebdding methods by a large margin, and were competitive with the best specialised word sense induction methods. The aim of the maximum likelihood clustering method's objective function is to learn to distinguish contexts in which the word of interest appears from those in which

it does not appear by learning multiple embeddings for the word; the fact that the method indeed learns multiple embeddings for polysemous words offers mathematical support for the intuition that the distribution of these words is better conceived of as multimodal than unimodal; the fact that the embeddings perform well on the word sense induction tasks suggests that a distributional model of word sense can produce results which agree with human intuitions.

I then proposed PolyLM, an end-to-end method for learning sense embeddings based on similar ideas to the maximum likelihood clustering method. I showed that it is simple to apply PolyLM not only to word sense induction, but to two other word sense-related natural language understanding tasks.

A strength of both approaches is that, given an instance of a word occurring in a context, they produce a probability distribution over the senses of the word for that context. This is a very useful form of output, and makes these methods easy to apply to downstream tasks.

One difficulty encountered was that when the number of sense embeddings assigned to a word is greater than its number of meanings, two or more embeddings corresponding to the same meaning may be learned. A preferable behaviour would be for only one embedding to be learned for each sense, and remaining embeddings to "die off," i.e. never be predicted to occur in any context. To combat this problem, we made the assumption that only one sense of a word ought to have a high probability of occurring in any given context. A term was added to the objective function of the MLCC method and PolyLM to reflect this assumption. The addition of this term lead to a large performance improvement on word sense induction.

A current weakness of both MLCC and PolyLM is that they assign a fixed number of senses to each word. In order to avoid missing senses of words which genuinely have many meanings, we guess a large number of senses per word, but this is computationally inefficient. Furthermore it seems that the distinctness loss does not entirely eliminate "duplicate" sense embeddings. A fruitful subject of future work might be to investigate how to dynamically add or remove word senses when required.

The experimental results provided some evidence that PolyLM has a better understanding of sense distinctions than MiniBERT, a similarly sized model belonging to the class of currently predominant contextualised word embeddings models. A question to explore in the future is whether this apparent advantage can be translated into better performance on general natural language understanding tasks such as question answering, dialogue generation and machine translation.

References

Eneko Agirre and Philip Edmonds. 2006. *Word Sense Disambiguation: Algorithms and Applications (Text, Speech and Language Technology)*. Springer-Verlag, Berlin, Heidelberg.

Felipe Almeida and Geraldo Xexéo. 2019. Word embeddings: A survey.

Reinald Kim Amplayo, Seung-won Hwang, and Min Song. 2019. AutoSense model for word sense induction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6212–6219.

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural biLM and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4867, Brussels, Belgium. Association for Computational Linguistics.

Asaf Amrami and Yoav Goldberg. 2019. Towards better substitution-based word sense induction.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019. An effective approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of*

the Association for Computational Linguistics, pages 194–203, Florence, Italy. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 130–138, Cadiz, Spain. PMLR.

Osman Başkaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 300–306, Atlanta, Georgia, USA. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2000. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense

embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Noam Chomsky. 1957. *Syntactic structures.* Janua linguarum. Series minor ; nr. 4. Mouton, s'Gravenhage.

Noam Chomsky. 1969. *Words and Objections Essays on the Work of W.V. Quine*, 1st ed. 1969.. edition, chapter Quine's Empirical Assumptions. D. Reidel, Dordrecht.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175.

Haim Dubossarsky, Eitan Grossman, and Daphna Weinshall. 2018. Coming to your senses: on controls and evaluation sets in polysemy research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1732–1740, Brussels, Belgium. Association for Computational Linguistics.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 406–414, New York, NY, USA. Association for Computing Machinery.

J. Firth. 1957. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*. Philological Society, Oxford. Reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow.

Ilya Flyamer. 2017. adjustText.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610. IJCNN 2005.

Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, pages 17–24.

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361.

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10(2-3):146–162.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea. Association for Computational Linguistics.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3507–3512, Hong Kong, China. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. 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 95–105, Beijing, China. Association for Computational Linguistics.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

David Jurgens and Ioannis Klapaftis. 2013. SemEval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical*

*and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299, Atlanta, Georgia, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Guang-He Lee and Yun-Nung Chen. 2017. MUSE: Modularizing unsupervised sense embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 327–337, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. SenseBERT: Driving some sense into BERT.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Loureiro and Alípio Jorge. 2019. Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disam-

biguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411, Brussels, Belgium. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 task 14: Word sense induction &disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 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, Baltimore, Maryland. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 419–426, Madison, WI, USA. Omnipress.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the*

Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Steven Neale. 2018. A survey on automatically-constructed WordNets and their evaluation: Lexical and word embedding-based approaches. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Alexander Panchenko. 2016. Best of both worlds: Making word sense embeddings interpretable. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2649–2655, Portorož, Slovenia. European Language Resources Association (ELRA).

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

Catia Pesquita, Daniel Faria, André O. Falcão, Phillip Lord, and Francisco M. Couto. 2009. Semantic similarity in biomedical ontologies. *PLOS Computational Biology*, 5(7):1–12.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1680–1690, Austin, Texas. Association for Computational Linguistics.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability theory and related fields*, 102(2):145–158.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-*

2007), pages 87–92, Prague, Czech Republic. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-dependent sense embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 183–191, Austin, Texas. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Sebastian Ruder, Anders Søgaard, and Ivan Vulić. 2019. Unsupervised cross-

lingual representation learning. In *Proceedings of ACL 2019, Tutorial Abstracts*, pages 31–38.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension.

C. E Shannon. 1951. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30(1):50–64.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, Haitao Mi, and Daniel Gildea. 2016. Sense embedding learning for word sense induction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides G.M. Petrakis, and Evangelos E. Milios. 2005. Semantic similarity methods in wordnet and their application to information retrieval on the web. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, WIDM '05, page 10–16, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2017. Sense embeddings in knowledge-based word sense disambiguation. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of WordNet for neural word sense disambiguation. In *Proceedings of the Tenth Global Wordnet Conference*, pages 108–117, Poland.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop Black-*

boxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. 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 323–333, Beijing, China. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 236–246, Berlin, Germany. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 19–27, Washington, DC, USA. IEEE Computer Society.

George K. Zipf. 1950. Human behavior and the principle of least effort. *Journal of Clinical Psychology*, 6(3):306–306.