Chapter 1

Finding Word Sense Embeddings Of Known Meaning

Abstract

Word sense embeddings are vector representations of polysemous words – words with multiple meanings. These induced sense embeddings, however, do not necessarily correspond to any dictionary senses of the word. To overcome this, we propose a method to find new sense embeddings with known meaning. We term this method refitting, as the new embedding is fitted to model the meaning of a target word in an example sentence. The new lexically refitted embeddings are learnt using the probabilities of the existing induced sense embeddings, as well as their vector values. Our contributions are threefold: (1) The refitting method to find the new sense embeddings; (2) a novel smoothing technique, for use with the refitting method; and (3) a new similarity measure for words in context, defined by using the refitted sense embeddings. We show how our techniques improve the performance of the Adaptive Skip-Gram sense embeddings for word similarly evaluation; and how they allow the embeddings to be used for lexical word sense disambiguation.

1.1 Introduction

Popular word embedding vectors, such as Word2Vec, represent a word's semantic meaning and its syntactic role as a point in a vector space (**mikolov2013efficient**; **pennington2014glove**). As each word is only given one embedding, such methods are restricted to the representation of only a single combined sense, or meaning, of the word. *Word sense embeddings* generalise word embeddings to handle polysemous and homonymous words. Often these sense embeddings are learnt through unsupervised Word Sense Induction (WSI) (**Reisinger2010**; **Huang2012**; **tian2014probabilistic**; **AdaGrams**). The induced sense embeddings are unlikely to directly coincide with any set of human defined meaning at all, i.e. they will not match lexical senses such as those defined in a lexical dictionary, e.g. WordNet (**miller1995wordnet**). These induced senses may be more specific, more broad, or include the meanings of jargon not in common use.

One may argue that WSI systems can capture better word senses than human lexicographers do manually. However, this does not mean that induced senses can replace standard lexical senses. It is important to appreciate the vast wealth of existing knowledge defined around lexical senses. Methods to link induced senses to lexical senses allow us to take advantage of both worlds.

We propose *a refitting method* to generate a sense embedding vector that matches with a labelled lexical sense. Given an example sentence with the labelled lexical sense of a particular word, the refitting method algorithmically combines the induced sense embeddings of the target word such that the likelihood of the example sentence is maximised. We find that in doing so, the sense of the word in that sentence is captured. With the refitting, the induced sense embeddings are now able to be used in more general situations where standard senses, or user defined senses are desired.

Refitting word sense vectors to match a lexicographical sense inventory, such as WordNet or a translator's dictionary, is possible if the sense inventory features at least one example of the target sense's use. Our method allows this to be done very rapidly, and from only the single example of use this has with possible applications in low-resource languages.

Refitting can also be used to fit to a user provided example, giving a specific sense vector for that use. This has strong applications in information retrieval. The user can provide an example of a use of the word they are interested in. For example, searching for documents about "`banks`" as

in “the river banks were very muddy”. By generating an embedding for that specific sense, and by comparing with the generated embeddings in the indexed documents, we can not only pick up on suitable uses of other-words for example “`beach`” and “`shore`”, but also exclude different usages, for example of a financial bank. The method we propose, using our refitted embeddings, has lower time complexity than AvgSimC (**Reisinger2010**), the current standard method for evaluating the similarity of words in context. This is detailed in Section 1.5.1.

We noted during refitting, that a single induced sense would often dominate the refitted representation. It is rare in natural language for the meaning to be so unequivocal. Generally, a significant overlap exists between the meaning of different lexical senses, and there is often a high level of disagreement when humans are asked to annotate a corpus (**veronis1998study**). We would expect that during refitting there would likewise be contention over the most likely induced sense. Towards this end, we develop a smoothing method, which we call *geometric smoothing* that de-emphasises the sharp decisions made by the (unsmoothed) refitting method. We found that this significantly improves the results. This suggests that the sharpness of sense decisions is an issue with the language model, which smoothing can correct. The geometric smoothing method is presented in Section 1.3.2.

We demonstrate the refitting method on sense embedding vectors induced using Adaptive Skip-Grams (AdaGram) (**AdaGrams**), as well as our own simple greedy word sense embeddings. The method is applicable to any skip-gram-like language model that can take a sense vector as its input, and can output the probability of a word appearing in that sense’s context.

The rest of the paper is organised as follows: Section 1.2 briefly discusses two areas of related works. Section 1.3 presents our refitting method, as well as our proposed geometric smoothing method. Section 1.4 describes the WSI embedding models used in the evaluations. Section 1.5 defines the RefittedSim measure for word similarity in context, and presents its results. Section 1.6 shows how the refitted sense vectors can be used for lexical WSD. Finally, the paper concludes in Section 1.7.

1.2 Related Works

1.2.1 Directly Learning Lexical Sense Embeddings

In this area of research, the induction of word sense embeddings is treated as a supervised, or semi-supervised task, that requires sense labelled corpora for training.

Iacobacci et al. (**iacobacci2015sensembed**) use a Continuous Bag of Word language model (**mikolov2013efficient**), using word senses as the labels rather than words. This is a direct application of word embedding techniques. To overcome the lack of a large sense labelled corpus, Iacobacci et al. use a 3rd party WSD tool, BabelFly (**Moro2014**), to add sense annotations to a previously unlabelled corpus.

Chen et al. (**Chen2014**) use a supervised approach to train sense vectors, with an unsupervised WSD labelling step. They partially disambiguate their training corpus, using word sense vectors based on WordNet; and use these labels to train their embeddings. This relabelled data is then used as training data, for finding sense embeddings using skip-grams.

Our refitting method learns a new sense embedding as a weighted sum of existing induced sense embeddings of the target word. Refitting is a one-shot learning solution, as compared to the approaches used in the works discussed above. A notable advantage is the time taken to add a new sense. Adding a new sense is practically instantaneous, and replacing the entire sense inventory, of several hundred thousand senses, is only a matter of a few hours. Whereas for the existing approaches this would require repeating the training process, which will often take several days. Refitting is a process done to word sense embeddings, rather than a method for finding sense embeddings from a large corpus.

1.2.2 Mapping induced senses to lexical senses

By defining a stochastic map between the induced and lexical senses, Agirre et al. (**agirre2006**), propose a general method for allowing WSI systems to be used for WSD. Their work was used in SemEval-2007 Task 02 (**SemEval2007WSIandWSD**) to evaluate all entries. Agirre et al. use a mapping corpus to find the probability of a lexical sense, given the induced sense according to the WSI system. This is more general than the approach we propose here, which only works for sense embedding based WSI. By exploiting the particular properties of sense embedding based WSI systems we propose a system that can better facilitate the use of this subset of WSI systems for WSD.

1.3 Proposed Refitting Framework

The key contribution of this work is to provide a way to synthesise a word sense embedding given only a single example sentence and a set of pretrained sense embedding vectors. We termed this *refitting* the sense vectors. By refitting the unsupervised vectors we define a new vector, that lines up with the specific meaning of the word from the example sentence.

This can be looked at as a one-shot learning problem, analogous to regression. The training of the induced sense, and of the language model, can be considered an unsupervised pre-training step. The new word sense embedding should give a high value for the likelihood of the example sentence, according to the language model. It should also generalise to give a high likelihood of other contexts where this word sense occurs.

We initially attempted to directly optimise the sense vector to predict the example. We applied the L-BFGS (**nocedal1980updating**) optimisation algorithm with the sense vector being the parameter being optimised over, and the objective being to maximise the probability of the example sentence according to the language model. This was found to generalise poorly, due to over-fitting, and to be very slow. Rather than a direct approach, we instead take inspiration from the locally linear relationship between meaning and vector position that has been demonstrated for word embeddings (**mikolov2013efficient**; **mikolovSkip**; **mikolov2013linguisticsubstructures**).

To refit the induced sense embeddings to a particular meaning of a word, we express that a new embedding as as a weighted combination of the induced sense vectors. The weight is determined by the probability of each induced sense given the context.

Given a collection of induced (unlabelled) embeddings $\mathbf{u} = u_1, ..., u_{n_u}$, and an example sentence $\mathbf{c} = w_1, ..., w_{n_c}$ we define a function $l(\mathbf{u} \mid \mathbf{c})$ which determines the refitted sense vector, from the unsupervised vectors and the context as:

$$l(\mathbf{u} \mid \mathbf{c}) = \sum_{\forall u_i \in \mathbf{u}} u_i P(u_i \mid \mathbf{c}) \tag{1.1}$$

Bayes' Theorem can be used to estimate the posterior predictive distribution $P(u_i \mid \mathbf{c})$.

Bengio et al. (**NPLM**) describe a similar method to Equation (1.1) for finding (single sense) word embeddings for words not found in their vocabulary. The formula they give is as per Equation (1.1), but summing over the entire vocabulary of words (rather than just $\mathbf{u}$).

1.3.1 A General WSD method

Using the language model and application of Bayes' theorem, we define a general word sense disambiguation method that can be used for refitting (Equation (1.1)), and for lexical word sense disambiguation (see Section 1.6). This is a standard approach of using Bayes' theorem (**tian2014probabilistic**; **AdaGrams**). We present it here for completeness.

The context is used to determine which sense is the most suitable for this use of the *target word* (the word being disambiguated). Let $\mathbf{s} = (s_1, ..., s_n)$, be the collection of senses for the target word[1].

Let $\mathbf{c} = (w_1, ..., w_{n_c})$ be a sequence of words making up the context of the target word. For example for the target word *kid*, the context could be $\mathbf{c} = (\ $ *wow the wool from the, is, so, soft, and, fluffy*), where *kid* is the central word taken from between *the* and *fluffy*.

For any particular sense, s_i, the multiple sense skip-gram language model can be used to find the probability of a word w_j occurring in the context: $P(w_j \mid s_i)$. By assuming the conditional independence of each word w_j in the context, given the sense embedding s_i, the probability of the context can be calculated:

$$P(\mathbf{c} \mid s_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j \mid s_i) \tag{1.2}$$

The correctness of the conditional independence assumption depends on the quality of the representation – the ideal sense representation would fully capture all information about the contexts it can appear in – thus the other contexts elements would not present any additional information, and so $P(w_a \mid w_b, s_i) = P(w_a \mid s_i)$. Given this, we have an estimate of $P(\mathbf{c} \mid s_i)$ which can be used to find $P(s_i \mid \mathbf{c})$. However, a false assumption of independence contributes towards overly sharp estimates of the posterior distribution **rosenfeld2000two**, which we seek to address in Section 1.3.2 with geometric smoothing.

[1] As this part of our method is used with both the unsupervised senses and the lexical senses, referred to as $\mathbf{u}$ and $\mathbf{l}$ respectively in other parts of the paper, here we use a general sense $\mathbf{s}$ to avoid confusion.

Bayes' Theorem is applied to this context likelihood function $P(\mathbf{c} \mid s_i)$ and a prior for the sense $P(s_i)$ to allow the posterior probability to be found:

$$P(s_i \mid \mathbf{c}) = \frac{P(\mathbf{c} \mid s_i)P(s_i)}{\sum_{s_j \in \mathbf{s}} P(\mathbf{c} \mid s_j)P(s_j)} \tag{1.3}$$

This is the probability of the sense given the context.

1.3.2 Geometric Smoothing for General WSD

During refitting, we note that often one induced sense would be calculated as having much higher probability of occurring than the others (according to Equation (1.3)). This level of certainty is not expected to occur in natural languages, ambiguity is almost always possible. To resolve such dominance problems, we propose a new *geometric smoothing* method. This is suitable for smoothing posterior probability estimates derived from products of conditionally independent likelihoods. It smooths the resulting distribution, by shifting all probabilities to be closer to the uniform distribution.

We hypothesize that the sharpness of probability estimates from Equation (1.3) is a result of data sparsity, and of a false independence assumption in Equation (1.2). This is well known to occur for n-gram language models **rosenfeld2000two**. Word-embeddings language models largely overcome the data sparsity problem due to weight sharing effects (**NPLM**). We suggest that the problem remains for word sense embeddings, where there are many more classes. Thus the training data must be split further between each sense than it was when split for each word. The power law distribution of word use (**zipf1949human**) is compounded by word senses within those used also following the a power law distribution (**Kilgarriff2004**). Rare senses are liable to over-fit to the few contexts they do occur in, and so give disproportionately high likelihoods to contexts that those are similar to. We propose to handle these issues through additional smoothing.

We consider replacing the unnormalised posterior with its n_c-th root, where n_c is the length of the context. We replace the likelihood of Equation (1.2) with $P_S(\mathbf{c} \mid s_i) = \prod_{\forall w_j \in \mathbf{c}} \sqrt[n_c]{P(w_j \mid s_i)}$. Similarly, we replace the prior with: $P_S(s_i) = \sqrt[n_c]{P(w_j \mid s_i)}$ When this is substituted into Equation (1.3), it becomes a smoothed version of $P(s_i \mid \mathbf{c})$.

$$P_S(s_i \mid \mathbf{c}) = \frac{\sqrt[n_c]{P(\mathbf{c} \mid s_i)P(s_i)}}{\sum_{s_j \in \mathbf{s}} \sqrt[n_c]{P(\mathbf{c} \mid s_j)P(s_j)}} \tag{1.4}$$

The motivation for taking the n_c-th root comes from considering the case of the uniform prior. In this case $P_S(\mathbf{c} \mid s_i)$ is the geometric mean of the individual word probabilities $P_S(w_j \mid s_i)$. Consider, if one has two context sentences, $\mathbf{c} = \{w_1, ..., w_{n_c}\}$ and $\mathbf{c}' = \{w'_1, ..., w'_{n_{c'}}\}$, such that $n'_c > n'_c$ then using Equation (1.2) to calculate $P(\mathbf{c} \mid s_i)$ and $P(\mathbf{c}' \mid s_i)$ will result in incomparable results as additional number of probability terms will dominate – often significantly more than the relative values of the probabilities themselves. The number of words that can occur in the context of any given sense is very large – a large portion of the vocabulary. We would expect, averaging across all words, that each addition word in the context would decrease the probability by a factor of $\frac{1}{V}$, where V is the vocabulary size. The expected probabilities for $P(\mathbf{c} \mid s_i)$ is $\frac{1}{V^{n_c}}$ and for $P(\mathbf{c}' \mid s_i)$ is $\frac{1}{V^{n_{c'}}}$. As $n_{c'} > n_c$, thus we expect $P(\mathbf{c}' \mid s_i) \ll P(\mathbf{c} \mid s_i)$. Taking the n_c-th and $n_{c'}$-th roots of $P(\mathbf{c} \mid s_i)$ and $P(\mathbf{c} \mid s_i)$ normalises these probabilities so that they have the same expected value; thus making a context-length independent comparison possible. When this normalisation is applied to Equation (1.3), we get the smoothing effect.

1.4 Experimental Sense Embedding Models

We trained two sense embedding models, AdaGram (**AdaGrams**) and our own Greedy Sense Embedding method. During training we use the Wikipedia dataset as used by Huang et al. (**Huang2012**). However, we do not perform the extensive preprocessing used in that work.

Most of our evaluations are carried out on Adaptive SkipGrams (AdaGram) (**AdaGrams**). AdaGram is a non-parametric Bayesian extension of Skip-gram. It learns a number of different word senses, as are required to properly model the language.

We use the implementation[2] provided by the authors with minor adjustments for Julia (**Julia**) v0.5 compatibility.

[2] `https://github.com/sbos/AdaGram.jl`

The AdaGram model was configured to have up to 30 senses per word, where each sense is represented by a 100 dimension vector. The sense threshold was set to 10^{-10} to encourage many senses. Only words with at least 20 occurrences are kept, this gives a total vocabulary size of 497,537 words.

To confirm that our techniques are not merely a quirk of the AdaGram method or its implementation, we implemented a new simple baseline word sense embedding method. This method starts with a fixed number of randomly initialised embeddings, then greedily assigns each training case to the sense which predicts it with the highest probability (using Equation (1.3)). The task remains the same: using skip-grams with hierarchical softmax to predict the context words for the input word sense. This is similar to **neelakantan2015efficient**, however it is using collocation probability, rather than distance in vector-space as the sense assignment measure. Our implementation is based on a heavily modified version of Word2Vec.jl[3].

This method is intrinsically worse than AdaGram. Nothing in the model encourages diversification and specialisation of the embeddings. Manual inspection reveals that a variety of senses are captured, though with significant repetition of common senses, and with rare senses being missed. Regardless of its low quality, it is a fully independent method from AdaGram, and so is suitable for our use in checking the generalisation of the refitting techniques.

The vocabulary used is smaller than for the AdaGram model. Words with at least 20,000 occurrences are allocated 20 senses. Words with at least 250 occurrences are restricted to a single sense. The remaining rare words are discarded. This results in a vocabulary size of 88,262, with 2,796 words having multiple senses. We always use a uniform prior, as the model does not facilitate easy calculation of the prior.

1.5 Similarity of Words in Context

Estimating word similarity with context is the task of determining how similar words are, when presented with the context they occur in. The goal of this task is to match human judgements of word similarity. For each of the target words and contexts; we use refitting on the target word to create a word sense embedding specialised for the meaning in the context provided. Then the similarity of the refitted vectors can be measured using cosine distance (or similar). By measuring similarity this way, we are defining a new similarity measure.

Reisinger and Mooney (**Reisinger2010**) define a number of measures for word similarity suitable for use with sense embeddings. The most successful was AvgSimC, which has become the gold standard method for use on similarity tasks. It has been used with great success in many works **Huang2012**; **Chen2014**; **tian2014probabilistic**.

AvgSimC is defined using distance metric d (normally cosine distance) as:

$$\text{AvgSimC}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) = \frac{1}{n \times n'} \sum_{u_i \in \mathbf{u}} \sum_{u'_j \in \mathbf{u}'} P(u_i \mid \mathbf{c})\, P(u'_j \mid \mathbf{c}')\, d(u_i, u'_j) \tag{1.5}$$

for contexts $\mathbf{c}$ and $\mathbf{c}'$, the contexts of the two words to be compared, and for $\mathbf{u} = \{u_1, ..., u_n\}$ and $\mathbf{u}' = \{u'_1, ..., u\prime_{n'}\}$ the respective sets of induced senses of the two words.

1.5.1 A New Similarity Measure: RefittedSim

We define a new similarity measure, RefittedSim, as the distance between the refitted sense embeddings. As shown in Figure 1.1 the example contexts are used to refit the induced sense embeddings of each word. This is a direct application of Equation (1.1).

Using the same definitions as in Equation (1.5), RefittedSim is defined as:

$$\text{RefittedSim}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) \;=\; d(l(\mathbf{u} \mid \mathbf{c}), l(\mathbf{u}' \mid \mathbf{c}') = d\left(\textstyle\sum_{u_i \in \mathbf{u}} u_i P(u_i \mid \mathbf{c}),\; \sum_{u'_j \in \mathbf{u}'} u_i P(u'_j \mid \mathbf{c}')\right) \tag{1.6}$$

AvgSimC is a probability weighted average of pairwise computed distances for each sense vector. Whereas RefittedSim is a single distance measured between the two refitted vectors – which are the probability weighted averages of the original unsupervised sense vectors.

There is a notable difference in time complexity between AvgSimC and RefittedSim. AvgSimC has time complexity $O(n \left\|\mathbf{c}\right\| + n' \left\|\mathbf{c}'\right\| + n \times n')$, while RefittedSim has $O(n \left\|\mathbf{c}\right\| + n' \left\|\mathbf{c}'\right\|)$. The product of the number of senses of each word $n \times n'$, may be small for dictionary senses, but it

[3] `https://github.com/tanmaykm/Word2Vec.jl/`

Figure 1.1: Block diagram for RefittedSim similarity measure

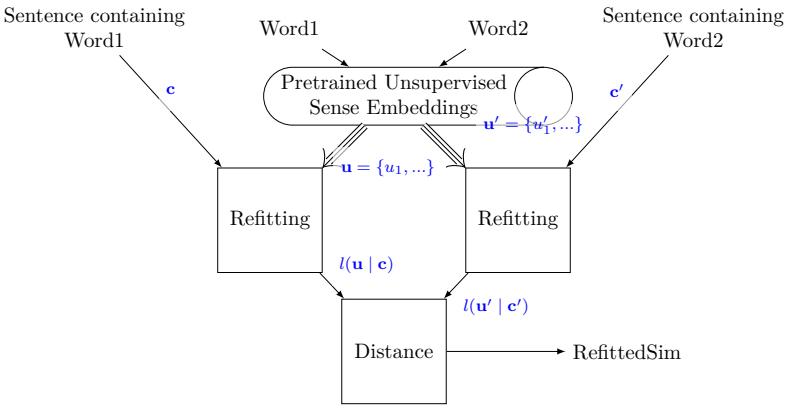

Table 1.1: Spearman rank correlation $\rho \times 100$ when evaluated on the SCWS task, for varying hyper-parameters.

Method	Geometric Smoothing	Use Prior	AvgSimC	RefittedSim
AdaGram	T	T	**53.8**	64.8
AdaGram	T	F	36.1	**65.0**
AdaGram	F	T	43.8	47.8
AdaGram	F	F	20.7	24.1
Greedy	T	F	23.6	49.7
Greedy	F	F	22.2	40.7

is often large for induced senses. Dictionaries tend to define only a few senses per word – the average[4] number of senses per word in WordNet is less than three (**miller1995wordnet**). For induced senses, however, it is often desirable to train many more senses, to get better results using the more fine-grained information. Reisinger and Mooney (**Reisinger2010**) found optimal results in several evaluations near 50 senses. In such cases the $O(n \times n')$ is significant, avoiding it with RefittedSim makes the similarity measure more useful for information retrieval.

1.5.2 Experimental Setup

We evaluate our refitting method using Stanford's Contextual Word Similarities (SCWS) dataset (**Huang2012**). During evaluation, each context paragraph is limited to 5 words to either side of the target word, as in the training.

1.5.3 Results

Table 1.1 shows the results of our evaluations on the SCWS similarity task. A significant improvement can be seen by applying our techniques.

[4]It should be noted, though, that the number of meanings is not normally distributed (**zipf1945meaning**).

Table 1.2: Spearman rank correlation $\rho \times 100$ when evaluated on the SCWS task, compared to other methods . RefittedSim-S is with smoothing, and RefittedSim-SU is with uniform prior

Paper	Embedding	Similarity	$\rho \times 100$
This paper	AdaGram	AvgSimC	43.8
This paper	AdaGram	RefittedSim-S	64.8
This paper	AdaGram	RefittedSim-SU	65.0
Huang2012	Huang et al.	AvgSimC	65.7
Huang2012	Pruned tf-idf	AvgSimC	60.5
Chen2014	Chen et al.	AvgSimC	68.9
tian2014probabilistic	Tian et al.	AvgSimC	65.4
tian2014probabilistic	Tian et al.	MaxSim	**65.6**
iacobacci2015sensembed	SenseEmbed	Min Tanimoto	58.9
iacobacci2015sensembed	SenseEmbed	Weighted Tanimoto	62.4

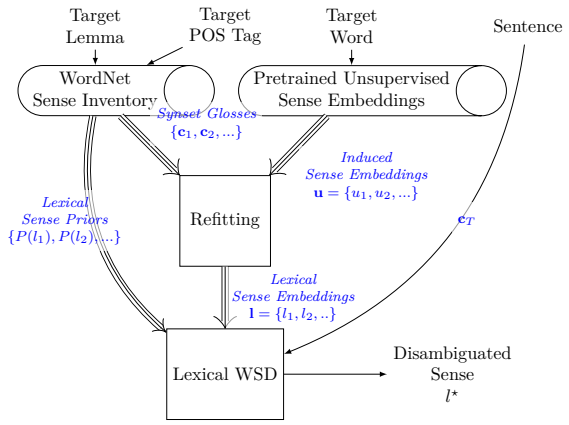

Figure 1.2: Block diagram for performing WSD using refitting.

The RefittedSim method consistently outperforms AvgSimC across all configurations. Similarly geometric smoothing consistently improves performance both for AvgSimC and for RefittedSim. The improvement is significantly more for RefittedSim than for AvgSimC results. In general using the unsupervised sense prior estimate from the AdaGram model, improves performance – particularly for AvgSimC. The exception to this is with RefittedSim with smoothing, where it makes very little difference. Unsurprisingly, given its low quality, the Greedy embeddings are always outperformed by AdaGram. It is not clear if these improvements will transfer to clustering based methods due to the differences in how the sense probability is estimated, compared to the language model based method evaluated on in Table 1.1.

Table 1.2 compares our results with those reported in the literature using other methods. These results are not directly comparable, as each method uses a different training corpus, with different preprocessing steps, which can have significant effects on performance. It can been seen that by applying our techniques we bring the results of our AdaGram model from very poor ($\rho \times 100 = 43.8$) when using normal AvgSimC without smoothing, up to being competitive with other models, when using RefittedSim with smoothing. The method of Chen et al. (**Chen2014**), has a significant lead on the other results presented. This can be attributed to its very effective semi-supervised fine-tuning method. This suggests a possible avenue for future development in using refitted sense vectors to relabel a corpus, and then performing fine-tuning similar to that done by Chen et al.

1.6 Word Sense Disambiguation

1.6.1 Refitting for Word Sense Disambiguation

Once refitting has been used to create sense vectors for lexical word senses, an obvious used of them is to perform word sense disambiguation. In this section we refer to the lexical word sense disambiguation problem, i.e. to take a word and find its dictionary sense; whereas the methods discussed in Equations (1.3) and (1.4) consider the more general problem, as applicable to disambiguating lexical or induced word senses depending on the inputs. Our overall process shown in Figure 1.2 uses both: first disambiguating the induced senses as part of refitting, then using the refitted sense vectors to find the most likely dictionary sense.

First, refitting is used to transform the induced sense vectors into lexical sense vectors. We use the targeted word's lemma (i.e. base form), and part of speech (POS) tag to retrieve all possible definitions of the word (Glosses) from WordNet; there is one gloss per sense. These glosses are used as the example sentence to perform refitting (see Section 1.3). We find embeddings, $\mathbf{l} = \{l_1, ..., l_{n_l}\}$ for each of the lexical word senses using Equation (1.1). These lexical word senses are still supported by the language model, which means one can apply the general WSD method to determine the posterior probability of a word sense, given an observed context.

When given a sentence $\mathbf{c}_T$, containing a target word to be disambiguated, the probability of each lexical word sense $P(l_i \mid \mathbf{c}_T)$, can be found using Equation (1.3) (or the smoothed version Equation (1.4)), over the lexically refitted sense embeddings. Then, selecting the correct sense is simply selecting the most likely sense:

$$l^\star(\mathbf{l}, \mathbf{c}_T) = \underset{\forall l_i \in \mathbf{l}}{\operatorname{argmax}} P(l_i|\mathbf{c}_T) = \underset{\forall l_i \in \mathbf{l}}{\operatorname{argmax}} \frac{P(\mathbf{c}_T \mid l_i)P(l_i)}{\sum_{\forall l_j \in \mathbf{l}} P(\mathbf{c}_T \mid l_j)P(l_j)} \tag{1.7}$$

Method	Attempted	Precision	Recall	F1
Refitted-S AdaGram	99.91%	0.799	0.799	**0.799**
Refitted AdaGram	99.91%	0.774	0.773	0.774
Refitted-S Greedy	79.95%	0.797	0.637	0.708
Refitted-S Greedy $*$	100.00%	0.793	0.793	0.793
Refitted Greedy	79.95%	0.725	0.580	0.645
Refitted Greedy $*$	100.00%	0.793	0.793	0.793
Mapped AdaGram	84.31%	0.776	0.654	0.710
Mapped AdaGram $*$	100.00%	0.736	0.736	0.736
MFS baseline	100.00%	0.789	0.789	0.789

Table 1.3: Results on SemEval 2007 Task 7 – course-all-words disambiguation. The -S marks results using geometric smoothing. The $*$ marks results with MSF backoff.

1.6.2 Lexical Sense Prior

WordNet includes frequency counts for each word sense based on Semcor (**tengi1998design**). These form a prior for $P(l_i)$. The comparatively small size of Semcor means that many word senses do not occur at all. We apply add-one smoothing to remove any zero counts. This is in addition to using our proposed geometric smoothing as an optional part of the general WSD. Geometric smoothing serves a different (but related) purpose, of decreasing the sharpness of the likelihood function – not of removing impossibilities from the prior.

1.6.3 Experimental Setup

The WSD performance is evaluated on the SemEval 2007 Task 7.

We use the weighted mapping method of Agirre et al. (**agirre2006**), (see Section 1.2.2) as a baseline alternative method for using WSI senses for WSD. We use Semcor as the mapping corpus, to derive the mapping weights.

The second baseline we use is the Most Frequent Sense (MFS). This method always disambiguates any word as having its most common meaning. Due to the power law distribution of word senses, this is a very effective heuristic (**Kilgarriff2004**). We also consider the results when using a backoff to MSF when a method is unable to determine the word sense the method can report the MFS instead of returning no result (a non-attempt).

1.6.4 Word Sense Disambiguation Results

The results of employing our method for WSD , are shown in Table 1.3. Our results using smoothed refitting, both with AdaGram and Greed Embeddings with backoff, outperform the MSF baseline **Navigli:2007:STC:1621474.1621480** – noted as a surprisingly hard baseline to beat (**Chen2014**).

The mapping method (**agirre2006**) was not up to the task of mapping unsupervised senses to supervised senses, on this large scale task. The Refitting method works better. Though refitting is only usable for language-model embedding WSI, the mapping method is suitable for all WSI systems.

While not directly comparable due to the difference in training data, we note that our Refitted results, are similar in performance, as measured by F1 score, to the results reported by Chen et al. (**Chen2014**). AdaGram with smoothing, and Greedy embeddings with backoff have close to the same result as reported for L2R with backoff – with the AdaGram slightly better and the Greedy embeddings slightly worse. They are exceeded by the best method reported in that paper: S2C method with backoff. Comparison to non-embedding based methods is not discussed here for brevity. Historically state of the art systems have functioned very differently; normally by approaching the WSD task by more direct means.

Our results are not strong enough for Refitted AdaGram to be used as a WSD method on its own, but do demonstrate that the senses found by refitting are capturing the information from lexical senses. It is now evident that the refitted sense embeddings are able to perform WSD, which was not possible with the unsupervised senses.

1.7 Conclusion

A new method is proposed for taking unsupervised word embeddings, and adapting them to align to particular given lexical senses, or user provided usage examples. This refitting method thus allows us to find word sense embeddings with known meaning. This method can be seen as a one-shot learning task, where only a single labelled example of each class is available for training. We show how our method can be used to create embeddings to evaluate the similarity of words, given their contexts.

This allows us to propose a new similarity measuring method, RefittedSim. The performance of RefittedSim on AdaGram is comparable to the results reported by the researchers of other sense embeddings techniques using AvgSimC, but its time complexity is significantly lower. We also demonstrate how similar refitting principles can be used to create a set of vectors that are aligned to the meanings in a sense inventory, such as WordNet.

We show how this can be used for word sense disambiguation. On this difficult task, it performs marginally better than the hard to beat MFS baseline, and significantly better than a general mapping method used for working with WSI senses on lexical WSD tasks. As part of our method for refitting, we present a geometric smoothing to overcome the issues of overly dominant senses probability estimates. We show that this significantly improves the performance. Our refitting method provides effective bridging between the vector space representation of meaning, and the traditional discrete lexical representation. More generally it allows a sense embedding to be created to model the meaning of a word in any given sentence. Significant applications of sense embeddings in tasks such as more accurate information retrieval thus become possible.