

A Method for Refitting Word Sense Embeddings Using a Single Example

ABSTRACT

Word sense embeddings are an extension of word embedding methods to handle polysemous words – words with multiple meanings. The induced word sense embeddings, however, do not correspond to any dictionary senses of the words. This limits the ability to integrate these systems with existing knowledge bases. To overcome this, we propose a method to find new word sense embeddings from single example sentences. We term this method refitting, as the new embedding is fitted to model the meaning in the example sentence using the existing unlabelled word sense embeddings.

Our contributions are threefold: The refitting method to find the new sense embeddings with known meanings, a novel smoothing technique, for use with the refitting method, and a new similarity measure for words in context, using these refitted sense embeddings.

We show how our refitting based techniques improve the performance of the Adaptive Skip-Gram word sense embeddings at word similarity evaluation; and how it allows the embeddings to be used for lexical word sense disambiguation – which was not possible before refitting the sense embeddings.

1. INTRODUCTION

Word embeddings represent a word’s semantic meaning and syntactic role in a vector space [1, 2, 3]. However, each word is only given one embeddings – which restricts it to representing only one (combined) meaning. Word sense embeddings are the generalisation of this to handle polysemous and homonymous words. Often these word sense embeddings are learned through unsupervised word sense induction [4, 5, 6, 7]. In these cases it is not possible to directly determine which meaning belongs to which embedding. Furthermore the induced senses are unlikely to directly match to any one human defined meaning at all; but rather be more fine, or more broad, or capture the meaning of particular jargon not in common use.

It can be argued that many word sense induction (WSI) systems may capture better senses than a human lexicogra-

phers, however this does not mean induced senses can replace standard lexical senses. While the induced senses may cover the space of meanings more comprehensively, or with better granularity than standard senses there is a vast wealth of existing knowledge built around various standard lexical senses. Methods to link induced senses to lexical senses, allow this information to be unlocked.

We propose a *refitting* method to allow induced word sense vectors to be converted to labelled word sense vectors, allowing them to be used with lexical knowledge bases. We show that this technique can be used to allow for word sense disambiguation to a lexical sense inventory – something that can not be done with the original induced sense vectors. We further demonstrate the usefulness of refitting, and this correctness of the results, by using it to define a new similarity measure for words with context which we call *RefittedSim*, this similarity measure is significantly faster than the commonly used *AvgSimC*. We noted that when refitting, often one induced sense overly dominated in finding the refitted sense. We developed a new smoothing method, *geometric smoothing*, suitable for smoothing these. We demonstrated its effectiveness at improving over earlier results.

We thus propose a method for using a single example of a word in context, to synthesise a new embedding for that particular sense. We term this *refitting* the induced senses, as it combines then to fit to the meaning in the example sentence. Our method allows us to use a single labelled example to produce a labelled sense embedding. This allows a shift from an representation that can only work within the system, to one that uses a standard sense, or a user defined sense, which can be used as part of a more complex knowledge engineering system. This refitting method has several applications.

One such use is to refit word sense vectors to a lexicographical sense inventory, such as WordNet, or from a translator’s dictionary – so long as the source features at least one example of use, or a definition. The new lexically refitted word sense can then be used for Word Sense Disambiguation (WSD). Applying WSD to a adds useful information to a unstructured document, allowing further processing methods to take advantage of this information. One particular application of this would be as part of a machine translation system. To properly translate a word, the correct word sense should be determined, as different word senses in the source language, often translate to entirely different words in the target language. Using WSD to add sense annotation to a document bridges between unstructured data and further more structured processing. Details on how our

method is used for WSD are in Section 6.1.

Beyond refitting to a standard lexical sense, refitting to a user provided example has applications in information retrieval. Consider the natural language query “Find me all webpages about banks as in ‘the river banks were very muddy.’”. By generating an embedding vector for that specific sense of “banks” from the example sentence, and generating one from use of the word in each retrieved document, we can approximate measuring distance in semantic space. This allows for similarity ranking – discarding irrelevant uses of the word. The method we propose, using our refitted embeddings, has lower time complexity than the current state of the art alternative. This is detailed in Section 5.1

Our refitting method synthesises a new word sense vector using the existing the induced word sense vectors and the language model. The new vector is formed as a appropriately weighted sum of the original embeddings. A weighting of the induced sense vectors is determined using the language model and the example sentence. The new vector approximately corresponds to the meaning given by the example sentence. The method is detailed in Section 3.

We noted that while our refitted word sense vectors did capture the meaning of the example sentence, often a single induced sense would dominate the refitted sense representation. We note that rarely in natural language is the meaning so clear cut. There is generally a significant overlap between the meaning of senses, and often a high level of disagreement when humans are asked to annotate a corpus [8]. We thus would expect this also to be true when automatically determining the appropriateness of the induced senses, during refitting. Towards this end, we develop a smoothing method, which we call *geometric smoothing* that emphasises the sharp decisions made by the (unsmoothed) refitting method. We found that this significantly improves results. This suggests that the sharpness of decisions from the language model may be an artefact of the training method and the sparseness of the training data, which smoothing can correct. The geometric smoothing method is presented in Section 3.2.

We demonstrate the refitting method on Adaptive Skip-Grams (AdaGram) [7], and on our own simple greedy multiple word-sense embeddings. The method is generally applicable to any skip-gram-like language model that can take a word sense vector as it’s input, and can output the probability of a word appearing in that word sense’s context.

The rest of the paper is organised as follows: Section 2 discusses related works on learning standard lexical sense repressions directly, and on associating the induced senses with standard lexical senses. Section 3 presents our refitting method, as well as the geometric smoothing method used with it. Section 6.1 shows how the refitted word sense vectors can be used for lexical word sense disambiguation. Section 5 describes the RefittedSim measure for word similarity in context, and presents its results. Finally, the paper presents it’s conclusions in Section 7.

2. RELATED WORKS


2.1 Directly Learning Lexical Sense Embeddings

Our refitting method can be considered as learning lexical sense embeddings, as a one-shot learning problem. The alternative is to transform the problem into a supervised or semi-supervised learning task. The difficulty lies in how to

get a sufficient number of labelled senses. One option is to use a separate WSD method to artificially label an unlabelled corpus.

Iacobacci et al. [9] use a Continuous Bag of Word (CBOW) [3] language model, but use 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 the 3rd party WSD tool BabelFly, to add sense annotations to a previously unlabelled corpus. An alternative approach would be to use a method that disambiguates the senses using a partially trained model.

Chen et al. use a semi-supervised approach to train word sense vectors [10]. They partially disambiguate their training corpus, using initial word sense vectors and WordNet; and use these labels to fine-tune their embeddings. Initial the word sense vectors are set as the average of the single sense word embeddings[3] for the words in the WordNet gloss. Similarly, they define a context vector, as the average of all words in a sentence. They then progressively label the words with the senses that for the sense vector that is closest to the context vector, if it is closer than a fixed threshold. The selected sense vector is used to update the context vector and then the next word is relabelled. They give two methods for selecting the order of relabelling. They then fine tune the sense embeddings by defining a new skip-gram method with the objective of predicting both the words and word-senses in the context, when given a input word¹. The requirement for meeting a threshold in order to add the sense label, decreases the likelihood of training on an incorrect sense label. The overall approach is the process the data add labels where confident, based on initial sense vectors estimated from the glosses, and then use these as the training target.

The key practical difference between the one-shot approach used by refitting, compared to the supervised, and semi-supervised approach uses in existing works it the time to retrain to add a new sense. With our refitting fitting approach added a new sense is practically instantiations, and replacing the entire sense inventory only a matter of hours. Whereas for the existing approaches adding senses would require retraining nearly from scratch. This is because refitting is a process done to sense word embeddings, rather a method for finding word-sense embeddings itself. It can be applied to any trained word sense embeddings, so long  meeting the requirement for a suitable language model.

One of the key reasons, that it is useful to have senses that aligned with standard lexical senses, is that it allows word sense embeddings to be used for word sense disambiguation. This is demonstrated by Chen et al. [10], and we discuss how refitted word sense vectors can do the same in Section 6.1. If the word senses are do not correspond to the lexical senses, then to use them for WSD requires forming an a mapping from induced senses, to the lexical senses.

2.2 Mapping induces senses to lexical senses

For senses that do not correspond to the senses defined by a lexicography, Agirre et al. gave a general method to use them for lexical WSD [11]. Their method applies to all word sense induction systems, not just word embedding systems, so is more general than the refitting approach we present.

¹Note that using the input word to predict the sense of words from the context is the opposite of the approach used by AdaGram. AdaGram uses the word-sense to predict the word of the context [7].

Their method maps the induced sense disambiguation scores, to scores for disambiguating standard lexical senses. This method was used for Semeval-2007 Task 02 [12] to evaluate all entries. They use a annotated *mapping corpus* to construct a mapping weight between induced senses probabilities and the lexical sense probabilities.

For $\mathbf{l} = \{l_1, \dots\}$ the set of lexically defined senses, $\mathbf{u} = u_1, \dots$ the set of unsupervised induced senses, and a corpus of labelled contexts \mathbf{c}_1, \dots Agirree et al. define the mapping weight by a likelihood. This likelihood $P(l_i | u_j)$ is estimated by counting how often u_j is the most-likely sense as returned by WSD on the induced senses, when l_i is the labelled lexical sense; and applying the definition of conditional probability.

This method requires the mapping corpus cover every lexical word sense, with enough instances for the law of large numbers to apply such that the frequency count converges to the actual likelihood.

From this when presented with a sentence containing the target word to disambiguate (\mathbf{c}) the unsupervised sense score ($P(u_j | \mathbf{c})$) are converted to supervised scores ($P(l_j | \mathbf{c})$) by

$$P(l_i | \mathbf{c}) = P(l_i | u_j) P(u_j | \mathbf{c}) \quad (1)$$

Agirree et al. demonstrate this method with the small Semeval 3 English Lexical Sample [13] – it is less clear how well it will work on more complete tasks featuring rarer words and senses. We use this method as a baseline in Section 6.1.

3. REFITTING FRAMEWORK

The key contribution of this work is to provide a way to synthesis a word sense embedding given only a single example. For lack of a better term we call this *refitting* the word 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. The training of the induced sense, and of the language model, can be considered the a unsupervised pre-training step. The new word sense embedding should give a high value for the likelihood of the example sentence, under the language model. Further more though, it should generalise to also give high likelihood of other contexts that were never seen, but which also occur near the word of this particular meaning.

We attempted directly optimising the sense vector as to maximise the probability of the example when input into the language model as part of preliminary investigations. This was found to generalise poorly, due to over-fitting. It also took a significant amount of time. Rather than a direct approach, we instead take inspiration from the famous locally linear relationship between meaning and vector position demonstrated with single sense word embeddings [3, 14, 15].

In order to refit the sense embedded to align to the meaning of the word in a particular context, we express it as a combination of the unsupervised sense vectors for that word. The new word sense vector is a weighted sum of the existing vectors that were already trained. Where the weight is determined by the probability of each induced sense given the context.

Given a collection of induced (unlabelled) embeddings $\mathbf{u} = u_1, \dots, u_{n_u}$, and example sentence which we will again call $\mathbf{c} = w_1, \dots, w_{n_c}$. We define a function $l(\mathbf{u} | \mathbf{c})$ which that determines the refitted sense vector, from the unsupervised vectors and the context.

$$l(\mathbf{u} | \mathbf{c}) = \sum_{\forall u_i \in \mathbf{u}} u_i P(u_i | \mathbf{c}) \quad (2)$$

To do this, we need to estimate the posterior predictive distribution $P(u_i | \mathbf{c})$. This can be done simply by using Bayes’ Theorem, as shown in Section 3.1. Further to that though, we present an alternative, for estimating a smoothed version of the result in Equation (7). We find in general that it is beneficial to smooth the distribution, to prevent a single sense dominating the sum.

In the very first neural network language model paper, Bengio et al. describe a similar method to Equation (2) for finding the word embeddings for words not found in their vocabulary [1]. They suggest that if a word was not in the training data, “an initial feature vector for such a word, by taking a weighted convex combination of the feature vectors of other words that could have occurred in the same context, with weights proportional to their conditional probability”. The formula they give is as per Equation (2), but summing over the entire vocabulary of words (rather than just \mathbf{u}). To the best of our knowledge, this method has not been used for handling out-of-vocabulary words in any more recent word embedding architectures, nor has it ever been used for word sense vectors.

3.0.1 Fallback for dictionary phrases (collocations)

Unless a specialised tokenizing method is used, a word embedding method will not learn embedding for collocations such as “civil war” or “martial arts”. Normal tokenizers will split them at the word level, learning embeddings for “civil”, and “war”, and for “martial” and “arts”. This issue is often considered minimal for word embeddings, as a approximate embedding can be constructed by summing embeddings for each word in the phrase.

For single-sense word embeddings the summing the word embeddings of the words making up the phrase in the phrase results in reasonable representation [14, 16]. As we are already creating a weighted sum, in the refitting step (Equation (2)), we can facilitate a similar result by adding the additional sense embeddings for each word to the total pool of sense embeddings to be combined (\mathbf{u} above). This allows for the senses of one word to contribute more than the other.

It is likely that one word in a collocation will contain senses with most specialised used for the collocation than the other. For example, “civil war”: “war” as in “civil war” appears in similar contexts to any other sense of “war”. But “civil” as in “civil war” appears in rather different contexts to the other uses of “civil”. Thus we expect there to be a sense embedding for “civil” that is particularly useful in refitting for “civil war”.

The extreme version of this is if one or more words in the collocation have no embedding defined. In this case we fall back to only using the embeddings from the remain words. An example of this would be “Fulton County Court”, while “County” and “Court” are common words, “Fulton” is a rare proper noun. We use the remaining words: “County” and “Court” to determine the meaning of the whole.

3.1 A General WSD method

Using the language model, and simple application of Bayes’ theorem, we define a general word sense disambiguation method that will be used for refitting (Equation (2)), it will also be used for lexical word sense disambiguation (see

Section 6.1). This is a standard approach of using Bayes’ theorem, it has been presented elsewhere, including in the including in the work of Tian et al. in this same context [6, 7]. We present it here by way of background.

Taking some collection of word sense representations, we aim to use the context to determine which sense is the most suitable for this use. We will call the word we are trying to disambiguate the *target word*. Let $\mathbf{s} = (s_1, \dots, s_n)$, be the collection of possible word sense representations for target word, they may be induced senses, or lexical senses. Let $\mathbf{c} = (w_1, \dots, w_m)$ be a sequence of word from around the target word – that is to say it’s context window. For example for the target word *kid*, the context would be $\mathbf{c} = (\text{wow}, \text{the}, \text{wool}, \text{from}, \text{the}, \text{is}, \text{so}, \text{soft}, \text{and}, \text{fluffy})$, where *kid* is the central word taken from between *the* and *fluffy*. Ideally our contexts would be symmetric with similar window size to that used for training the language model, though this is not always possible.

For any particular word 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: $P(w_j | s_i)$ [6, 7]. By assuming conditional independence of each word w_j in the context, given then sense embedding s_i , the probability of the context given the sense can be calculated using language model:

$$P(\mathbf{c} | s_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j | s_i) \quad (3)$$

The correctness of the assumption of conditional independence depends on the quality of the representation – the ideal sense representation would fully capture all information about the contexts it can appear in – thus making the other elements of those contexts not present any additional information, and so $P(w_a | w_b, s_i) = P(w_a | s_i)$. Given this, we have a good estimate of $P(\mathbf{c} | s_i)$ which can be used with Bayes’s theorem to find $P(s_i | \mathbf{c})$.

Bayes’ Theorem can be applied to this context likelihood function $P(\mathbf{c} | s_i)$ and a prior for the sense $P(s_i)$, allows posterior probability to be found.

$$P(s_i | \mathbf{c}) = \frac{P_S(\mathbf{c} | s_i)P(s_i)}{\sum_{s_j \in \mathbf{s}} P_S(s_j | \mathbf{c})P(s_j)} \quad (4)$$

This is the probability of the sense, given the context.

Note that in an implementation of this in software, it is important to work with the logarithms of probabilities, to avoid numeric underflow. As is common practice when working with products of probabilities [press2007numerical].

We also propose an smoothed version of Equation (4).

3.2 Geometric Smoothing for General WSD

We propose a new smoothing function that we can apply to the general WSD sub-step of refitting. We call this method geometric smoothing. Geometric smoothing 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 – more-so the further they are from it.

It was noted that during refitting, often as single induced sense would be evaluated by Equation (4), as much more likely than the others. This level of certainty is not expected to occur in natural language. Consider sentences such as

“The CEO of the bank, went for a picnic by the river.” While “CEO” is closely linked to a financial bank, and “river” is strongly linked to a river bank, we do not wish for the occurrence of either word in the context to hard negate the possibility of either sense – this use of bank does refer to a financial institution, but there are other sentences with very similar words in the context window that refer to a river bank. By using a smoothing method we investigate if this sharp focus on a sense was causing issues. Our results in Section 6.1 and Section 3 confirm that indeed smoothing the sense probability estimates does improved performance.

We posit that the sharpness of probability estimates from Equation (4), are the result of a data sparsity problem. While word sense embeddings largely overcome the data sparsity problem that plagued ngram languages models through weight sharing effects Bengio, Ducharme, Vincent, and Janvin [1], we suggest data sparsity continues to cause problems for multiple word sense embeddings. Multiple sense word embeddings have a significantly larger set of labels to learn; the same amount of training data must be shared amongst many times more senses, than it would per word. Further to this, not only do words in a corpus occur with frequency according to a power law distribution [17], the relative frequency of word senses are also distributed according to a power law [18]. The rare (induced) word senses are thus extremely rare. Thus these rare word senses, will over-fit to the few contexts they do occur in, and so give disproportionately high likelihoods to contexts that they are similar to. From the other side: as the training examples have to be shared amongst each word sense, rarer words are unlikely to only occur in the context of more than one sense – thus causing the other senses to predict that it is unlikely to occur in their context. Weight sharing effects do go a way to offsetting this, but we suggest the data sparsity problem can become too great with word sense embeddings. Thus requiring the additional smoothing.

In geometric smoothing we consider instead replacing the, with it’s n_c -th root. Where n_c is the length of the context.

$$P_S(\mathbf{c} | s_i) = \prod_{\forall w_j \in \mathbf{c}} \sqrt[n_c]{P(w_j | s_i)} \quad (5)$$

Similarly, we replace the prior (if not uniform) with its n_c -th root

$$P_S(s_i) = \sqrt[n_c]{P(s_i)} \quad (6)$$

When this is substituted into Equation (4), it smooths $P(s_i | \mathbf{c})$.

$$\begin{aligned} P(s_i | \mathbf{c}) &= \frac{P_S(\mathbf{c} | s_i)P(s_i)}{\sum_{s_j \in \mathbf{s}} P_S(s_j | \mathbf{c})P(s_j)} \\ &= \frac{\sqrt[n_c]{P(\mathbf{c} | s_i)P(s_i)}}{\sum_{s_j \in \mathbf{s}} \sqrt[n_c]{P(s_j | \mathbf{c})P(s_j)}} \end{aligned} \quad (7)$$

The motivation for this comes from considering the case of the uniform prior. In this case, it is the same as replacing $P_S(\mathbf{c} | s_i)$ with the geometric mean of the individual word probabilities $P_S(w_j | s_i)$. If one has two sentences, $\mathbf{c} = \{w_1, \dots, w_{|\mathbf{c}|}\}$ and $\mathbf{c}' = \{w'_1, \dots, w'_{|\mathbf{c}'|}\}$, such that $|\mathbf{c}'| > |\mathbf{c}|$; then using Equation (3) to calculate $P(\mathbf{c} | s_i)$ and $P(\mathbf{c}' | s_i)$ will generally result in incomparable results as addition probability terms will dominate – often significantly more than the relative values of the probabilities themselves.

This is because for almost any word w and any target word sense $P(w \mid s_i)$ the number of words that could be in the targets context is a significant portion of the entire vocabulary. This becomes clear when one considers the expected values. For V the vocabulary size, we have the expected value:

$$\mathbb{E}_{\mathbf{c}}(P(\mathbf{c} \mid s_i)) = (\mathbb{E}_w(P(w \mid s_i)))^{|\mathbf{c}|} = \frac{1}{V^{|\mathbf{c}|}} \quad (8)$$

Taking the $|\mathbf{c}|$ -th and $|\mathbf{c}'|$ -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 fair comparison possible, where the context length does not dominate. When this normalisation is applied to Equation (4), we get a smoothing effect.

4. METHOD

4.1 A Baseline Greedy Word Sense Embedding Method

To confirm that our refitting method, and associated techniques are not simply a quirk of the AdaGram method or implementation, we defined a new simple baseline word sense embedding method. This method starts with a fixed number of randomly initialised word-sense embeddings, then at each training step greedily aligns each training cases to the sense for which predicts that context with highest probability (using Equation (4)). The task remains the same: using skip-grams with hierarchical softmax to predict context words for the word sense. Our implementation is based on a heavily modified version of the Word2Vec.jl package by Tanmay Mohapatra, and Zhixuan Yang ² for word embeddings.

Due to the greedy nature of this baseline method, it is a intrinsically worse than AdaGram. A particular embedding may get an initial lead at predicting a context, simply based on predicting high probability for all words. It then gets trained more, resulting in it generally predicting high probability of many words, while other embeddings remain untrained. There is no force in the model to encourage diversification and specialisation of the embeddings. As a greedy method it readily falls into traps where the most used embedding is the most trained embedding, and thus is likely to receive more training. The random initialisation does help with this. Manual inspection reveals that it does capture a variety of senses, though with significant repetition of common senses, and with rare senses being largely missed. It is however a fully independent method from AdaGram, and so is suitable for use in checking the generalisation of our method.

4.2 Experimental Setup and Model Parameters

During training we use the Wikipedia dataset as used by Huang et al. [5]. We did not perform the extensive preprocessing used in that work, We preprocessed the data merely by converting it to lower case, tokenizing it and removing punctuation tokens. For both models, were trained with a single iteration over the whole data set. Also in both cases sub-sampling of 10^{-5} was used, and decreasing learning rate starting at 0.25 was used.

4.2.1 AdaGram

The AdaGram model was configured to have up to 30 word senses, where each sense 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 were kept, this gave a total vocabulary size of 497,537 words.

We use the AdaGram [7] implementation³ provided by Bartunov et al. with minor adjustments for Julia [19] v0.5 compatibility.

4.3 Greedy Baseline Model

The greedy word sense embedding model, was configured without significant thought for it's performance – indeed it was selected to have significantly different parameters to the AdaGram model that we trained. So that impact of using our methods could be checked to see they performed consistently on different word sense embedding models.

Each sense embedding had 300 dimensions. The vocabulary was restricted to only words with at least 250 occurrences, which resulted in a total vocabulary of 88,262 words. Words with at least 20,000 occurrences, were giving 20 senses, and the remainder just a single sense. This resulted in the most common 2,796 words having multiple senses. This is not very high coverage, however it is more substantial than may be expected as the most common words having the most word senses [20], and being vastly more common than the less common words [17, 21].

With these greedy embeddings, we always use the uniform prior, as the assignment of contexts embeddings can change significantly during training, and to determinate an accurate estimation of the prior would take similar time to performing an full iterator of training.

5. SIMILARITY OF WORD 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 the task is the match human judgements of word similarity.

For each of the target words and contexts; we can 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 take takes advantage of context – distinct from the commonly used AvgSimC.

5.1 RefittedSim


RefittedSim is simply defined by using the refitting to determine a suitable word sense vector for the context of each work to be measured, and then measuring the distance between the refitted sense embeddings. This is a direct application of Equation (2).

For \mathbf{c} and \mathbf{c}' the contexts of target words w and w' for which we want to measure the similarity of. Where w has senses $\mathbf{s} = \{s_1, \dots, s_n\}$, and w' has senses $\mathbf{s}' = \{s'_1, \dots, s'_{n'}\}$, and for d the distance function – normally cosine distance. We define the RefittedSim as

²<https://github.com/tanmaykm/Word2Vec.jl/>

³<https://github.com/sbos/AdaGram.jl>

$$\begin{aligned}
& \text{RefittedSim}((\mathbf{s}, \mathbf{c}), (\mathbf{s}', \mathbf{c}')) \\
&= d(l(\mathbf{s} | \mathbf{c}), l(\mathbf{s}' | \mathbf{c}')) \\
&= d \sum_{s_i \in \mathbf{s}} s_i P(s_i | \mathbf{c}), \sum_{s'_j \in \mathbf{s}'} s'_j P(s'_j | \mathbf{c}') \quad (9)
\end{aligned}$$

 We can contrast this against the commonly used AvgSimC, as proposed by Reisinger and Mooney [4].

$$\begin{aligned}
& \text{AvgSimC}((\mathbf{s}, \mathbf{c}), (\mathbf{s}', \mathbf{c}')) \\
&= \frac{1}{n \times n'} \sum_{s_i \in \mathbf{s}} \sum_{s'_j \in \mathbf{s}'} P(s_i | \mathbf{c}) P(s'_j | \mathbf{c}') d(s_i, s'_j) \quad (10)
\end{aligned}$$

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

AvgSimC is used in multiple existing works. It should be noted that many of these $P(s_i | \mathbf{c})$ and $P(s'_j | \mathbf{c}')$ using a probabilistic language model based method, but rather define them using the distance function [4, 5]. Context vector clustering based sense embeddings interpret the distance from their sense cluster centroid, as a as we can see it as a soft membership function with that sense. We note that this has a different distribution to that of which comes directly out of the skip-gram language model.

There is a notable time complexity difference between AvgSimC and RefittedSim. AvgSimC has time complexity $O(n \|\mathbf{c}\| + n' \|\mathbf{c}'\| + n \times n')$ RefittedSim is $O(n \|\mathbf{c}\| + n' \|\mathbf{c}'\|)$. The product of the number of senses of each word $n \times n'$, may small for dictionary senses, but it is often large for induced senses. Dictionaries tend to define only a few sense per word – the average⁴ number of senses per word in WordNet is less than three for all parts of speech [22]. For induced sense, however, it is often desirable to train many more senses, to get better results using the more fine-grained information. In several evaluations performed by Reisinger and Mooney they found optimal results at near to 50 senses [4]; We found similar results with our own preliminary experiments also. In the case of fine grained induced senses, the $O(n \times n')$ is significant; avoiding it with RefittedSim makes the similarity measure more useful for information retrieval.


It may be desired for retrieving relevant search results to allows the user to provide an example of the term being searched for to allow removal of relevant uses of the word in a different sense. consider for example a search for “Apple” as in ‘the fruit I like to eat’, can return different results from “Apple” as in ‘the company that makes the iPod, and the Macbook’. This is the practical usecase for similarity with context. In the information retrieval context, the probabilities of the word sense for the context of the document can be done off-line, during indexing. With this assumption, the query-time time complexity becomes: for AvgSimC becomes $O(n \times n')$, and for RefittedSim it becomes $O(1)$. We do note however that pre-computing the word sense probabilities for each word in the document during indexing remains expensive (though no more so than for AvgSimC). We suggest that when this indexing time is considered worthwhile, then

⁴It should be noted, though, that the number of meanings is not normally distributed [20]

Method	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

Table 1: Spearman rank correlation $\rho \times 100$, for various configurations of AdaGram and greedy Sense embeddings, when evaluated on the SCWS task.

Paper	Embedding	Similarity	$\rho \times 100$
This paper	AdaGram	AvgSimC	43.8
	AdaGram (with smoothing)	RefittedSim	64.8
	AdaGram (with smoothing and uniform prior)	RefittedSim	65.0
[5]	Huang et al.	AvgSimC	65.7
	Prunedtf-idf	AvgSimC	60.5
[10]	Chen et al.	AvgSimC	68.9
[6]	Tian et al.	AvgSimC	65.4
	Tian et al.	MaxSim	65.6
[9]	SenseEmbed	Min Tanimoto Distance	58.9
	SenseEmbed	Weighted Tanimoto Distances	62.4


 **Table 2: Spearman rank correlation $\rho \times 100$ as reported by several methods**

RefittedSim is significantly more viable for use in information retrieve tasks than AvgSimC. For many uses through, ignoring word senses in information retrieval continues to be viable, as documents can be differentiated by adding more query terms – this limits the practical use of both AvgSimC and RefittedSim.

5.2 Experimental Setup

We evaluate our refitting method using the Stanford’s Contextual Word Similarities (SCWS) corpus [5]. During evaluation each context was converted to lower case, and limited to 5 words to either side of the target word, as in the training. We also evaluated using the whole context, the results were found, in all cases, to be very similar, but the processing time much longer. For brevity, we have excluded those results from the tables presented below.

5.3 Results

Table 1 shows the results of evaluation on the SCWS similarity task, as different parameters are used or not. The table only shows our results, on multiple sense embeddings using language models. These improvements are not demonstrated to apply to clustering based model (these have not been evaluated as part of this study). It can be noted that RefittedSim constantly out performs AvgSimC. Geometric Smoothing consistently improves performance, for both AvgSimC and for RefittedSim. 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 Greeding embeddings are always outperformed by AdaGrams.

It is not clear as to if these improvements will transfor 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.

For greater context, Table 2 shows the our results compared to the results reported by other methods in the literature.

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 be seen though that by apply our techniques we bring the results of AdaGram (initially using AvgSimC without smoothing) from poor, up to being competitive with other model. The state of the art by Chen et al, is the only method with a significant lead [10]. This can be attributed to its very effective semi-supervised fine-tuning method. This suggest interesting future applications in using refitted sense vectors for relabelling a corpus, and then performing fine tuning similar to that done by Chen et al.

6. WORD SENSE DISAMBIGUATION

6.1 Refitting for Word Sense Disambiguation

Once refitting has been used to create word sense vectors that are matched to lexical word senses, we would like to use them to perform word sense disambiguation. This would show that the refitted embeddings are capturing the lexical information. In this section we refer to the standard word sense disambiguation problem – to take a word and find it’s dictionary sense; where-as Equation (4) and Equation (7) consider the more general word sense disambiguation problem, as applicable to disambiguate to lexical word senses, or induces word senses, depending on the inputs. Our overall process must to both: first disambiguating to induces senses as part of refitting, then using the refitted word sense vectors to find the most likely dictionary sense.

First we perform refitting, to transform the induces sense vectors, into lexical word sense vectors. Each sense of the word has a example sentence from the sense inventory – in the case of WordNet this is the gloss (i.e. the definition). By applying refitting to the unsupervised senses using

We initially re We use the Given a example sentence $\{\mathbf{c}_1, \dots, \mathbf{c}_{n_l}\}$, such as a gloss, for each lexical word sense we can refitted word sense vectors for each of $\mathbf{l} = \{l_1, \dots, l_{n_l}\}$ by using the method described in eq. (2).

Then when given target word w_{target} in a sentence \mathbf{c}_T to be disambiguated, we can apply Equation (4) (or the smoothed version Equation (7)), again to find $P(l_i | \mathbf{c}_T)$ for each lexical word sense of w using the lexically refitted sense vectors we found earlier.

$$l^* = \arg \max_{\forall l_i \in \mathbf{l}} P(l_i | \mathbf{c}_T) = \arg \max_{\forall l_i \in \mathbf{l}} \frac{P_s(\mathbf{c}_T | l_i) P(l_i)}{\sum_{\forall l_j \in \mathbf{l}} P_s(\mathbf{c}_T | l_j) P(l_j)}$$

As each lexical word-sense shares it’s gloss with the rest of its set of synonymous senses, this means that the lexically aligned word senses for synonym are all fitted to the same example sentence. However this does not mean they are equal, as they use different word sense vectors to find the weighted sums. Similarly, as the glosses are defined per lemma (base form), the different tenses (and other variants) of a word also share the same example sentence, but once again, the refitted word sense vector will still be different, so we learn different word sense vectors for each form of the word.

Note that in this case, we do have a prior for $P(l_i)$. WordNet includes frequency counts for each word sense based on Semcor Tengi [23]. However Semcor is not a immense corpus, being only a subset of the Brown corpus.

The comparatively small size of Semcor means that many word senses do not occur at all. As counted by WordNet 2.1 (used in the WSD task in ??), there are counts for just 36973 word senses, out of the total 207,016 senses in WordNet; i.e. 82% of all word senses have no count information.

There is an additional issue that the Semcor’s underling texts from Brown are now significantly ageing being all from 1961 [24] – it is not unreasonable to suggest that frequency of word-sense use has changed significantly in the last half century.

Never-the-less, the word count is the best prior readily available. Given the the highly unbalanced distribution of sense occurrence (as discussed in ??), a uniform prior would not be a reasonable approximation.

6.2 Word Sense Disambiguation Results

As discussed in Section 6.1 the language model, can be applied to a WSD Task by fitting to the glosses. We do not window the glosses as they are already short, and the actual word we are fitting may not occur in the gloss itself, being that it is a definition shared by mean words with the same meaning. We do convert the gloss to lower case, and strip any out of vocabulary words, including punctuation.

The results of employing our method, are shown in Table 3, with several others for comparison. Our results marginally outperforms the baseline most frequents sense – as often observed, a surprisingly difficult baseline to beat. We note, that it is particularly important to include the prior based on sense counts – to do otherwise results in incorrect formulation of Bayes theorem in Section 6.1. In this case, the uniform prior is snot a good approximation to the actual prior distribution – give highly unbalanced frequencies of different word senses.

Our method, like that of Chen et al only uses information from WordNet, and a unlabelled corpus [10]. It does not use hand-engineered features, or labelled training data – beyond the glosses from WordNet.

Our method is beaten by the S2C method, when used with MFS backoff. We do not employ any back-off strategy with AdaGram, as almost all words are found in the trained vocabulary. We suggest that threshold approach used with S2C is particularly effective at determining whether there is enough useful context information to make it clear as to if the word can be disambiguated, using the model.

When we compare our method, Refitted AdaGram, to Mapped Adagram which uses the method of Agirre et al [11]; which we implemented. Using both the Brown1 and Brown2 subcorpora of SemCor as the mapping corpus. It can be seen that the mapping method does not perform well largely due to the number words not found in the mapped vocabulary – but even when a backoff method is added to handle those cases, we see the results are still worse than MFS alone, and worse than the Refitted AdaGram. This indicates that the lack of data prevented the good estimation of the Mapping probabilities. Refitted AdaGram did not have this problem due to only needed a single example of each lexical word sense.

Even the very best methods for WSD do not perform substantially better than the baseline. It has been suggested that a ensemble method as the way forward to improve WSD beyond this level [26, 27]. We suggest that a method such as ours, based on unsupervised word embeddings, would be well suited to use in an ensemble, as it operates using very

Paper	Method	Attempted	Precision	Recall	F1	Backoff	Geometric Smoothing	Smooth priors
this paper	Refitted AdaGram	99.91%	0.783	0.783	0.783		T	F
	Refitted AdaGram	99.91%	0.799	0.799	0.799		T	T
	Refitted AdaGram	99.91%	0.774	0.773	0.774		F	F
	Refitted AdaGram	100%	0.799	0.799	0.799	MSF	T	T
	Refitted Greedy	79.95%	0.784	0.627	0.697		T	F
	Refitted Greedy	100%	0.783	0.783	0.783	MSF	T	F
	Refitted Greedy	79.95%	0.797	0.637	0.708		T	T
	Refitted Greedy	100%	0.793	0.793	0.793	MSF	T	T
	Mapped [11] AdaGram	84.31%	0.776	0.654	0.710		T	T
	Mapped [11] AdaGram	100%	0.777	0.777	0.777	MSF	T	T
	Mapped [11] AdaGram	99.91%	0.784	0.784	0.784	RF	T	T
	Mapped [11] AdaGram	100%	0.784	0.784	0.784	RF+MSF	T	T
[25]	NUS-PT*	100%	0.825	0.825	0.825	MSF		
	MFS baseline	100%	0.789	0.789	0.789	Is MFS		
[10]	L2R				0.739			
	S2C				0.758			
	L2R				0.796	MSF		
	S2C				0.826	MSF		

Table 3: Results on SemEval 2007 Task 7 – course-all-words disambiguation. For comparison we include subset of the results from the other indicated papers.

different features, to more complex WSD systems employing semantic relations.

7. CONCLUSION

We have presented a new method for taking unsupervised word embeddings, and adapting them to align to particular given uses. 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 showed how our method could be used create embeddings to evaluate the similarity of words, given there contexts. This itself is a new method, which has time complexity of $O(1)$ vs the time complexity of commonly used AvgSimC which is $O(n \times n')$. The performance of our method is comparable to AvgSimC.

We also demonstrated how similar principles, could be used to create a set of vectors that are aligned to the meanings in a sense inventory, such as WordNet. We showed how this could be used for word sense disambiguation. On this difficult task, it performed marginally better than the MFS baseline, and better than some implementations comparable systems. However, on it's own it we do not believe the method is strong enough for commercial use. Future similar systems, building upon our work would have strong uses in machine translation.

As part of our method for refitting the sense embeddings to there new senses, we presented a novel smoothing algorithm – geometric smoothing. This smoothing method, is required to compensate for the data sparsity problem that is intrinsic in learning senses from usage data. Even more so than the data sparsity problem for words which is normally solved by word embeddings, due to their being many more meanings for words than there are words, particularly when fine grained senses are employed. We show as part of the discussed methods that our smoothing gives substantial improvement over the results with unsmoothed language model.

References

- [1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The journal of machine learning research*:137–186, 2003.
- [2] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning*. ACM, 2008, pp. 160–167.
- [3] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Arxiv preprint arxiv:1301.3781*, 2013.
- [4] J. Reisinger and R. J. Mooney. Multi-prototype vector-space models of word meaning. In *Human language technologies: the 2010 annual conference of the north american chapter of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 109–117.
- [5] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th annual meeting of the association for computational linguistics: long papers-volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.
- [6] F. Tian, H. Dai, J. Bian, B. Gao, R. Zhang, E. Chen, and T.-Y. Liu. A probabilistic model for learning multi-prototype word embeddings. In *Coling*, 2014, pp. 151–160.
- [7] S. Bartunov, D. Kondrashkin, A. Osokin, and D. P. Vetrov. Breaking sticks and ambiguities with adaptive skip-gram. *Corr*, abs/1502.07257, 2015.
- [8] J. Véronis. A study of polysemy judgements and inter-annotator agreement. In *Programme and advanced papers of the senseval workshop*, 1998, pp. 2–4.

- [9] I. Iacobacci, M. T. Pilehvar, and R. Navigli. Sensembed: learning sense embeddings for word and relational similarity. In *Proceedings of acl*, 2015, pp. 95–105.
- [10] X. Chen, Z. Liu, and M. Sun. A unified model for word sense representation and disambiguation. In *Emnlp*. Citeseer, 2014, pp. 1025–1035.
- [11] E. Agirre, D. Martínez, O. L. De Lacalle, and A. Soroa. Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In *Proceedings of the first workshop on graph based methods for natural language processing*. Association for Computational Linguistics, 2006, pp. 89–96.
- [12] E. Agirre and A. Soroa. Semeval-2007 task 02: evaluating word sense induction and discrimination systems. In *Proceedings of the 4th international workshop on semantic evaluations*. In SemEval '07. Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 7–12.
- [13] R. Mihalcea, T. A. Chklovski, and A. Kilgarrieff. The senseval-3 english lexical sample task. In. Association for Computational Linguistics, 2004.
- [14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [15] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Hlt-naacl*, 2013, pp. 746–751.
- [16] L. White, R. Togneri, W. Liu, and M. Bennamoun. How well sentence embeddings capture meaning. In *Proceedings of the 20th australasian document computing symposium*. In ADCS '15. ACM, Parramatta, NSW, Australia, 2015, 9:1–9:8. ISBN: 978-1-4503-4040-3.
- [17] G. Zipf. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press, 1949.
- [18] A. Kilgarrieff. *How dominant is the commonest sense of a word?* In. *Text, speech and dialogue: 7th international conference, tsd 2004, brno, czech republic, september 8-11, 2004. proceedings*. P. Sojka, I. Kopeček, and K. Pala, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 103–111. ISBN: 978-3-540-30120-2.
- [19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: a fresh approach to numerical computing, 2014. arXiv: 1411.1607 [cs.MS].
- [20] G. K. Zipf. The meaning-frequency relationship of words. *The journal of general psychology*, 33(2):251–256, 1945.
- [21] S. Gilmour and M. Dras. Understanding the pheromone system within ant colony optimization. In, *Ai 2005: advances in artificial intelligence*, pp. 786–789. Springer, 2005.
- [22] G. A. Miller. Wordnet: a lexical database for english. *Communications of the acm*, 38(11):39–41, 1995.
- [23] R. I. Teng. *Wordnet: an electronic lexical database, the mit press, cambridge, massachusetts*. In. C. r. Fellbaum, editor, 1998. part Design and implementation of the WordNet lexical database and searching software, p. 105.
- [24] W. N. Francis and H. Kucera. Brown corpus manual. *Brown university*, 1979.
- [25] R. Navigli, K. C. Litkowski, and O. Hargraves. Semeval-2007 task 07: coarse-grained english all-words task. In *Proceedings of the 4th international workshop on semantic evaluations*. In SemEval '07. Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 30–35.
- [26] H. M. Saarikoski and S. Legrand. Building an optimal wsd ensemble using per-word selection of best system. In *Iberoamerican congress on pattern recognition*. Springer, 2006, pp. 864–872.
- [27] H. M. Saarikoski, S. Legrand, and A. Gelbukh. Defining classifier regions for wsd ensembles using word space features. In *Mexican international conference on artificial intelligence*. Springer, 2006, pp. 855–867.