Finding Word Sense Embeddings Of Known Meaning

Lyndon White
University of Western Australia
lyndon.white@research.uwa.edu.au

Roberto Togneri
University of Western Australia
roberto.togneri@uwa.edu.au

Wei Liu
University of Western Australia
wei.liu@uwa.edu.au

Mohammed Bennamoun
University of Western Australia
mohammed.bennamoun@uwa.edu.au

ABSTRACT

Word sense embeddings are vector representations of polysemous words – words with multiple meanings. Similar techniques to those used to find word embedding vectors, such as in Word2Vec, are often used to learn multiple sense embeddings for each word. These induced sense embeddings, however, do not necessarily correspond to any dictionary senses of the word. This limits the applicability of these embeddings in traditional semantic-orientated tasks such as lexical word sense disambiguation. To overcome this, we propose a method to find new sense embeddings of known meaning. We term this method refitting, as the new embedding is fitted to model the meaning of a target word in the example sentence. The refitting is accomplished using 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 these refitted sense embeddings.

We show how our refitting based techniques improve the performance of the Adaptive Skip-Gram sense embeddings for word similarly evaluation; and how it allows the embeddings to be used for lexical word sense disambiguation – which was not possible before refitting the sense embeddings. Further to this, the similarity measure we derive from the refitting has significantly better time-complexity than the commonly used AvgSimC; making this more a more suitable method for large scale web-scale datasets.

1. INTRODUCTION

Popular word embedding vectors, such as Word2Vec [1] and GLoVE [2], represent a word's semantic meaning and its syntactic role as a point in a vector space. As each word is only given one embedding such methods are restricted to only representing a single combined sense, or meaning, of the word. *Word sense embeddings* are the generalisation of word embeddings to handle polysemous and homonymous words. Often these sense embeddings are learnt through unsupervised word sense induction [3]–[6]. 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, eg WordNet [7]. These induced senses can be more specific, more broad, or include the meanings of jargon not in common use.

It can be argued that many word sense induction (WSI) systems may capture better word senses than human lexicographers do manually, however this does not mean that induced senses can replace standard lexical senses. While the induced senses may cover the space of meanings more comprehensively, or with better granularity than the lexical senses, it is important to appreciate the vast wealth of existing knowledge defined around lexical senses from various sense inventories. Methods to link induced senses to lexical senses allow us to take advantage of both worlds.

In this paper, 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 combined the induced sense embeddings of the target word in a way such that the likelihood of the example sentence, is similar usages, is maximised. With the refitting, the induced sense embeddings are not restricted to be used only in the system they are obtained from, but can also 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 senses use. The new lexically refitted sense embedding can then be used for Word Sense Disambiguation (WSD). Applying WSD is almost indispensable in any unstructured document understanding. For example, a machine translation system: to properly translate a word, the correct sense should be determined, as different senses in the source language, often translate to entirely different words in the target language. We demonstrate how the refitting method can be successfully applied for WSD in Section 6.

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 web pages about 'banks' as in 'the river banks were very`

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

muddy.'". By generating an embedding vector for that specific sense of "banks" from the example sentence, and by comparing with the generated one from use of the same word in each retrieved document, we can more accurately measure their distance in a 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 AvgSimC [3], the current state of the art alternative for measuring similarity of words in context. This is detailed in Section 5.2.

We noted during refitting, that a single induced sense would often dominate a word sense's 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 [8]. 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 decisions from the language model is an issue with the language model, which smoothing can correct. The geometric smoothing method is presented in Section 3.3.

We demonstrate the refitting method on sense embedding vectors induced using Adaptive Skip-Grams (AdaGram) [6], 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 2 discusses two areas of related works, one on directly learning standard lexical sense repressions, and the other on indirectly associating induced senses with standard lexical senses. Section 3 presents our refitting method, as well as the geometric smoothing method used with it. Section 5 describes the RefittedSim measure for word similarity in context, and presents its results. Section 6 shows how the refitted sense vectors can be used for lexical word sense disambiguation. Finally, the paper concludes with outlook to future works in Section 7.

2. RELATED WORKS

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. [9] use a Continuous Bag of Word language model [1], 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, to add sense annotations to a previously unlabelled corpus.

Chen et al. [10] use a semi-supervised approach to train sense vectors. They partially disambiguate their training corpus, using initial word sense vectors and WordNet; and use these labels to fine-turn their embeddings. Initially the sense vectors are set as the average of the single sense word embeddings [1] for the words in the WordNet gloss. Similarly, a context vector is defined as the average of all words in a sentence. They then progressively label the words with the sense vector that is closest to the context vector if it is closer than a predefined threshold. The requirement for meeting a threshold in order to add the sense label decreases the likelihood of training on an incorrect sense label. 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. After the relabelling is complete they fine tune the sense embeddings by defining a new skip-gram method with the objective of predicting both the words and the word senses in the context, when given an input word[1]. The overall approach is to use the initial sense vectors estimated from the glosses to add sense labels to some of the words, and then use these as the training target.

Our refitting method learns a new sense embedding as a weighted sum of existing induced sense embeddings of the target word. This is analogous to regression learning or one-shot learning using refitting. The key practical difference between the one-shot approach used by refitting, as compared to the supervised and semi-supervised approaches used in existing works, is the time taken to add a new sense. Adding a new sense using our approach is practically instantiations, and replacing the entire sense inventory, of several hundred thousand senses, is only a matter of a few hours. Whereas for the existing approaches adding senses would require almost fully repeating the training process, often taking several days. Refitting is a process done to sense word embeddings, rather a method for finding sense embeddings from a large corpus. It can be applied to any pretrained sense embeddings, so long as it uses a language model based approach.

2.2 Mapping induced senses to lexical senses

Rather than learning lexical senses directly, and alternative approach is to form a weighted mapping from induced senses to the lexical senses. Agirre et al. [11] give a general method to allow induced word senses to be used for lexical WSD. Their method applies to all word sense induction systems, not just word embedding systems, so is more general than the refitting approach that we present. Their method maps the induced sense disambiguation scores, to scores for disambiguating standard lexical senses. This method was used in SemEval-2007 Task 02 [12] to evaluate all entries. They use an annotated *mapping corpus* to construct a mapping weight between induced senses probabilities and the lexical sense probabilities.

For $\mathbf{l} = \{l_1, ..., l_m\}$ the set of lexically defined senses, $\mathbf{u} = \{u_1, ..., u_n\}$ the set of unsupervised induced senses, and a corpus of labelled contexts $\{\mathbf{c}_1, ..., \mathbf{c}_k\}$, Agirre et al. define the mapping weight by a likelihood. This likelihood $P(l_i \mid u_j)$ 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 to cover every lexical word sense, with enough instances for the frequency count to converge to the actual likelihood.

This is used to disambiguate a target word within a sentence ($\mathbf{c}_T$) by converting the unsupervised sense scores,

[1]Note that using the input word to predict the which word senses will appear in the context is the opposite of the approach used by AdaGram. AdaGram uses the word sense to predict the word of the context [6].

$P(u_j \mid \mathbf{c}_T)$, to supervised sense scores, $P(l_j \mid \mathbf{c}_T)$, by

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

Agirre et al. demonstrate this method with the small SensEval 3 English Lexical Sample [13]. However, it is less clear how well this method will work on more complete tasks featuring rarer words and senses.

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. Furthermore, it should generalise to also give a high likelihood of other contexts that were never seen, but which also occur near the word sense of this particular meaning.

As part of our preliminary investigations, we attempted directly optimising the sense vector to predict the example. We applied the L-BFGS [14] 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 the the language model. This was found to generalise poorly, due to over-fitting. It also took a significant amount of time per word sense to be fitted. Rather than a direct approach, we instead take inspiration from the locally linear relationship between meaning and vector position that has been demonstrated for (single sense) word embeddings [1], [15], [16].

In order to refit the sense embedding 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 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.

Assume we are 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:

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

To do this, we need to estimate the posterior predictive distribution $P(u_i \mid \mathbf{c})$. This can be done by using Bayes' Theorem, as shown in Section 3.2; or with our smoothed variation discussed in Section 3.3

In the very first neural network language model paper, Bengio et al. [17] describe a similar method to Equation (2) for finding the word embeddings for words not found in their vocabulary. They suggest that if a word was not present in the training data, they form "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" [17]. 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 sense vectors.

3.1 Fall back for Dictionary Phrases (Collocations)

Unless a specialised tokenizing method is used, a word embedding method will not learn embeddings for collocations such as "`civil war`" or "`martial arts`". Normal tokenizers will split these 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 an approximate embedding can be constructed by summing embeddings for each word in the phrase.

For single-sense word embeddings summing the word embeddings of the words making up the phrase results in reasonable representation [15], [18]. 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 more specialised use in the collocation than the other. For example, "`war`" as in "`civil war`" appears in similar contexts to any other senses 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 remaining 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.

3.2 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 (2)), and also can be used for lexical word sense disambiguation (see Section 6). This is a standard approach of using Bayes' theorem [5], [6]. We present it here for completeness.

Taking some collection of 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 $\mathbf{s} = (s_1, ..., s_n)$, be the collection of possible senses for the target word[2].

Let $\mathbf{c} = (w_1, ..., w_{n_c})$ be a sequence of words from around the target word – its context window. 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*. Ideally our context windows would be symmetric with similar window size to that used for training the language model, though this is not always possible.

[2]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.

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: $P(w_j \mid s_i)$ [5], [6]. By assuming the conditional independence of each word w_j in the context, given the sense embedding s_i, the probability of the context given the sense can be calculated using the language model:

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

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 making the other elements of those contexts 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 a good estimate of $P(\mathbf{c} \mid s_i)$ which can be used with Bayes' theorem to find $P(s_i \mid \mathbf{c})$. However, it is known that a false assumption of independence contributes towards overly sharp estimates of the posterior distribution [19], which we seek to address in Section 3.3 with geometric smoothing.

Bayes' Theorem can be 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:

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

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

Note that in a software implementation of these equations it is important to work with the logarithms of the probabilities to avoid numeric underflow; as is common practice when working with products of probabilities [20].

Further to Equation (4), we also developed a method, for estimating a smoothed version of the posterior predictive distribution.

3.3 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 (4)). This level of certainty is not expected to occur in natural languages. 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 completely 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 would refer to a river bank.

To resolve such dominance problems, we propose a new *geometric smoothing* function applying to the general WSD equation (Equation (4)). We posit that this geometric smoothing function 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 away they are from being uniform.

By using a smoothing method we investigate if this dominance of a sense is causing issues. As shown in Section 5 and Section 6, it is confirmed that smoothing the sense probability estimates does improve performance.

We hypothesize that the sharpness of probability estimates from Equation (4) is a result of data sparsity, and of a false independence assumption in Equation (3). False independence and training data sparsity cause overly sharp posterior distribution estimates, the is particularly a problem for n-gram language models [19]. Word embeddings largely overcome the data sparsity problem due to weight sharing effects [17], and by being a higher quality language model allow for the approximation of conditional independence of the words in the context. We posit that these problems remain for word sense embeddings, where there is a significant larger number of classes. Thus the training data must be split further between each sense than it was when split for each word. Further to this, the frequency of words [21] and word senses [22] within a corpus both follow a approximate power law distribution (Zipf's Law). Thus raw word senses are extremely rare. These rare senses, are liable to over-fit to the few contexts they do occur in, and so give disproportionately high likelihoods to contexts that they are similar to. We propose to handle these language model issues through additional smoothing.

In the proposed geometric smoothing, we consider instead 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 (3) with:

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

Similarly, we replace the prior with:

$$P_S(s_i) = \sqrt[n_c]{P(w_j \mid s_i)} \tag{6}$$

When this is substituted into Equation (4), it becomes a smoothed version of $P(s_i \mid \mathbf{c})$.

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

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 (3) to calculate $P(\mathbf{c} \mid s_i)$ and $P(\mathbf{c}' \mid s_i)$ will generally result in incomparable results as additional number of probability terms will dominate – often significantly more than the 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. Further, when this normalisation is applied to Equation (4), we get a smoothing effect. Thus handling our issues with overly sharp posterior estimates.

4. EXPERIMENTAL SENSE EMBEDDING MODELS

We trained two sense embedding models, one based on AdaGram [6] and the other uses our own Greedy Sense Embedding method. The induces sense embeddings from these two models are used to evaluate the performance of our methods on similarity in context (Section 5) and at word sense disambiguation (Section 6). For consistency these two methods were trained with the same data.

During training we use the Wikipedia dataset as used by Huang et al. [4]. However, we do not perform the extensive preprocessing used in that work. Only tokenization, the removal of all punctuation and the conversion of the raw text to lower case. Both the AdaGram and the Greedy models were trained with a single iteration over the whole data set. In both cases, sub-sampling of 10^{-5}, and a decreasing learning rate starting at 0.25 is used.

4.1 AdaGram

Most of our evaluations are carried out on Adaptive Skip-Grams (AdaGram) [6]. 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 AdaGram implementation[3] provided by Bartunov et al. [6] with minor adjustments for Julia [23] v0.5 compatibility.

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.

4.2 Greedy Word Sense Embeddings

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 sense embeddings, then at each training step greedily assigns each training case to the sense which predicts that context with the highest probability (using Equation (4)). The task remains the same: using skip-grams with hierarchical softmax to predict context words for the input word sense. Our implementation is based on a heavily modified version of the Word2Vec.jl[4] package by Tanmay Mohapatra, and Zhixuan Yang for word embeddings.

Due to the greedy nature of this baseline method, it is intrinsically worse than AdaGram. A particular embedding may get an initial lead at predicting a context. It then gets trained more, resulting in it generally predicting a high probability of many words. While other embeddings may remain untrained, and may be stuck in part of the vector space that does not correspond to predicting any context that will ever occur. Nothing in the model encourages 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. 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.

[3]https://github.com/sbos/AdaGram.jl

[4]https://github.com/tanmaykm/Word2Vec.jl/

Sense embeddings of 300 dimensions are used. The vocabulary is restricted to only words with at least 250 occurrences, which results in a vocabulary size of 88,262. Words with at least 20,000 occurrences, are giving 20 senses, and the remainder just a single sense. This results in the most common 2,796 words having multiple senses. This is not a near-full coverage of the language.

With these greedy embeddings, we always use a uniform prior, as the assignment of contexts to senses can change significantly during training, and to determine an accurate estimate of the prior would take similar amount of time as to perform another full iteration of training.

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 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, which competes with the commonly used AvgSimC.

5.1 AvgSimC

In their seminal work on sense vectors representations Reisinger and Mooney define a number of measures for word similarity suitable for use with sense embeddings [3]. 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 [4], [5], [10].

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

$$\begin{aligned}&\text{AvgSimC}((\mathbf{u},\mathbf{c}),(\mathbf{u}',\mathbf{c}'))\\&\quad=\frac{1}{n\times n'}\sum_{u_i\in\mathbf{u}}\sum_{u'_j\in\mathbf{u}'}P(u_i\mid\mathbf{c})\,P(u'_j\mid\mathbf{c}')\,d(u_i,u'_j)\end{aligned}\quad(8)$$

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 $\mathbf{u}'=\{u'_1,...,u'_{n'}\}$ the respective sets of induced senses of the two words.

It should be noted that some methods using AvgSimC do not define $P(s_i\mid\mathbf{c})$ and $P(s'_j\mid\mathbf{c}')$ with a probabilistic language model based method, but rather define them using a distance function [3], [4]. Context-vector-clustering based sense embeddings interpret the distance from their sense cluster centroid to the context vector, as a soft membership function with that sense's cluster. We note that this has a different distribution to that of which comes directly out of a skip-gram language model.

5.2 A New Similarity Measure: RefittedSim

We define a new similarity measure RefittedSim, by measuring the distance between the refitted sense embeddings. As shown in Figure 1 the example contexts are used to refit the induced sense embeddings of each word. This is a direct application of Equation (2).

Using the same definitions as in Equation (8), RefittedSim

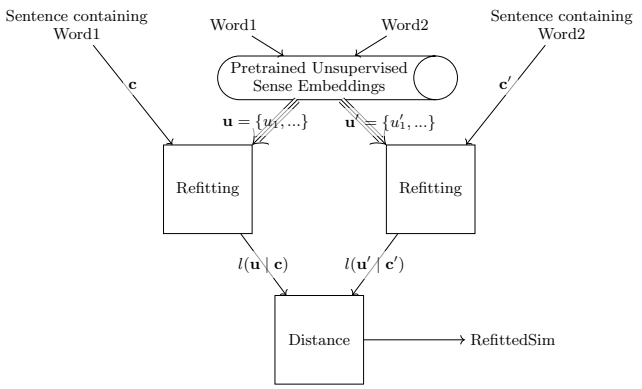

Figure 1: Block diagram for RefittedSim similarity measure

is defined as:

$$\begin{aligned}\text{RefittedSim}((\mathbf{u},\mathbf{c}),(\mathbf{u}',\mathbf{c}')) &= d(l(\mathbf{u}\mid\mathbf{c}),l(\mathbf{u}'\mid\mathbf{c}') \\ &= d\left(\sum_{u_i\in\mathbf{u}} u_i P(u_i\mid\mathbf{c}),\ \sum_{u'_j\in\mathbf{u}'} u_i P(u'_j\mid\mathbf{c}')\right) \end{aligned} \quad (9)$$

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

There is a notable difference in time complexity between AvgSimC and RefittedSim. AvgSimC has time complexity $O(n\,\|\mathbf{c}\| + n'\,\|\mathbf{c}'\| + n\times n')$, while RefittedSim has $O(n\,\|\mathbf{c}\| + n'\,\|\mathbf{c}'\|)$. The product of the number of senses of each word $n\times n'$, may be small for dictionary senses, but it is often large for induced senses. Dictionaries tend to define only a few senses per word – the average[5] number of senses per word in WordNet is less than three for all parts of speech [7]. For induced senses, 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 50 senses [3]. We found similar results with our own preliminary experiments. 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 desirable for retrieving relevant search results to allow the user to provide an example of the term being searched for. This would allow documents containing only uses of irrelevant meanings of the word can be filtered, while including documents containing effective synonyms. 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`". Alternatively, the same technique can be used to present results to the user grouped by the sense of their query terms. These are the practical use-case 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 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). If this indexing time is considered worthwhile, then RefittedSim is significantly more viable for use in information retrieval tasks than AvgSimC.

[5]It should be noted, though, that the number of meanings is not normally distributed [24].

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

Table 1: Spearman's rank correlation $\rho\times 100$, for various configurations of AgaGram and Greedy sense embeddings, when evaluated on the SCWS task.

5.3 Experimental Setup

We evaluate our refitting method using Stanford's Contextual Word Similarities (SCWS) dataset [4]. During evaluation each context paragraph is converted to lower case, and limited to 5 words to either side of the target word, as in the training.

5.4 Results

Table 1 shows the results of our evaluations on the SCWS similarity task. A significant improvement can be seen by applying our techniques.

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.

Table 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. [10], 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.

6. WORD SENSE DISAMBIGUATION

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
[4]	Huang et al.	AvgSimC	65.7
[4]	Pruned tf-idf	AvgSimC	60.5
[10]	Chen et al.	AvgSimC	**68.9**
[5]	Tian et al.	AvgSimC	65.4
[5]	Tian et al.	MaxSim	65.6
[9]	SenseEmbed	Min Tanimoto	58.9
[9]	SenseEmbed	Weighted Tanimoto	62.4

Table 2: Spearman rank correlation $\rho \times 100$ as reported by several methods. In this table RefittedSim-S refers to our RefittedSim using smoothing and the AdaGram prior, and SU to using smoothing and a uniform prior. AvgSimC is the original AvgSimC without smoothing but with the AdaGram prior.

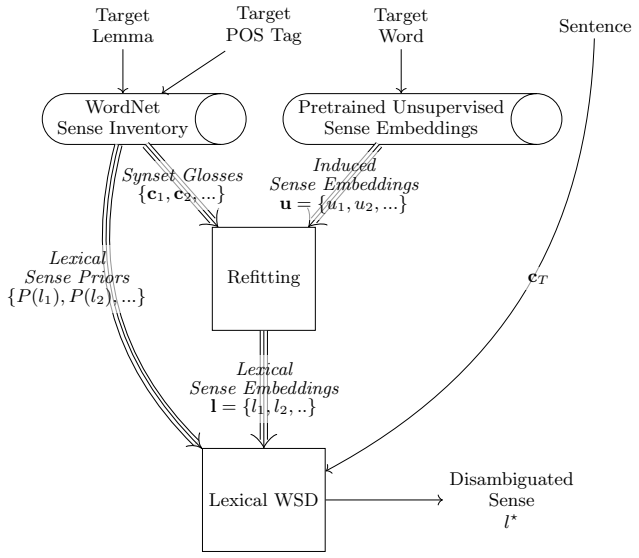

Figure 2: Block diagram for performing WSD using refitting

6.1 Refitting for Word Sense Disambiguation

Once refitting has been used to create sense vectors that are matched to lexical word senses, we would like to use them to perform word sense disambiguation. This would show whether or not the refitted embeddings are capturing the lexical information. 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 (4) and (7) consider the more general word sense disambiguation problem, as applicable to disambiguating lexical or induced word senses depending on the inputs. Our overall process must use both: first disambiguating the induced senses as part of refitting, then using the refitted sense vectors to find the most likely dictionary sense. The overall process is shown in Figure 2.

First, we perform refitting, 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. By applying refitting to the unsupervised senses using these example sentence, as discussed in Section 3, we find embeddings, $\mathbf{l} = \{l_1, ..., l_{n_l}\}$ for each of the lexical word senses using Equation (2). 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 (4) (or the smoothed version Equation (7)), over the lexically refitted sense embeddings. Then selecting the correct sense is simply selecting the most likely sense:

$$\begin{aligned} l^\star(\mathbf{l}, \mathbf{c}_T) &= \arg\max_{\forall l_i \in \mathbf{l}} P(l_i|\mathbf{c}_T) \\ &= \arg\max_{\forall l_i \in \mathbf{l}} \frac{P(\mathbf{c}_T \mid l_i)P(l_i)}{\sum_{\forall l_j \in \mathbf{l}} P(\mathbf{c}_T \mid l_j)P(l_j)} \end{aligned} \tag{10}$$

WordNet glosses are less than ideal examples sentences. As well as being definitions, rather than examples of use, they are shared across many words. Each lexical word sense shares its gloss with the rest of the set of synonymous senses from different words (synsets). Similarly as the synsets are defined per lemma (base word), the different lexeme (tenses and other variants) of a word also share the same example sentence. However, in both cases this does not mean that their refitted sense vectors are equal (though they are likely to be similar). When refitting, the different lexemes correspond to different induced sense vectors. As the induced sense vectors are contributing factors to the refitting sum, the refitted sense vectors are not identical even for the same example sentence (gloss). The limitations of WordNet glosses does not prevent our refitting system from functioning.

6.2 Lexical Sense Prior

WordNet includes frequency counts for each word sense based on Semcor Tengi [25]. These form a prior for $P(l_i)$.

However, Semcor is not an 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, there are counts for just 36,973 word senses, out of a total 207,016 senses in WordNet; i.e. 82% of all senses have no count information.

An additional issue is that Semcor's underling texts from Brown are now significantly ageing. They are all from 1961 [26] – it is not unreasonable to suggest that the 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 highly unbalanced distribution of sense occurrence a uniform prior would not be a reasonable approximation. We apply add-one smoothing to find the prior, 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.

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 [27]	100.00%	0.789	0.789	0.789

Table 3: Results on SemEval 2007 Task 7 – course-all-words disambiguation. The *-S* marks results using geometric smoothing. The ∗ marks results with MSF backoff.

6.3 Experimental Setup

The WSD performance is evaluated on the SemEval 2007 Task 7. WordNet 2.1, is used as the sense inventory. All glosses are converted to lower case, when used as the example sentences in the refitting step. They are not clipped to a window around the target word, as the target word often does not occur at all in the gloss; and the glosses are already close to the correct size.

We use the weighted mapping method of Agirre et al [11], (see Section 2.2) as a baseline alternative method for using WSI senses for WSD. When estimating the sense mapping weights we used both of the all-words-annotated subcorpora (Brown1 and Brown2) of SemCor as the mapping corpus. While calculating the weights for the map, we do clip to a 10 word context window for each target word to be disambiguated.

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 an effective heuristic [22].

We also evaluated the performance of the mapping baseline, and the Greedy embedding method, with a backoff to the 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). For embedding methods, this occurs when a polysemous word has only one (or zero) sense embeddings trained. For the mapping method it occurs when the word does not occur in the mapping corpus. We do not report the results for AdaGram with MSF backoff, as it was trained with a large enough vocabulary, that it has almost complete coverage.

6.4 Word Sense Disambiguation Results

The results of employing our method for WSD , are shown in Table 3. Our results with AdaGram when using refitting with geometric smoothing, outperform the MSF baseline – noted as a surprisingly hard baseline to beat [10]. Our results with the Greedy Embeddings, when using MSF backoff also exceed this baseline.

We found that the mapping method [11] was not up to the task of mapping unsupervised senses to supervised senses, on this large scale task. There is simply not enough data in SemCor to allow it to properly estimate the mapping weights. The Refitting method worked significantly better. Though refitting is only usable for language model based sense embeddings, whereas 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 [10]. 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.

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.

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 use 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 one lexical WSD tasks.

As part of our method for refitting the sense embeddings to their new senses, we present a geometric smoothing overcome the issues presented by the overly dominant senses probabilities estimates caused by limited training data. We show that this improves the results for the similarity task with both RefittedSim, and with AvgSimC; and also improves the WSD results.

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 a word in any given sentence. Significant applications of sense embeddings in tasks such as more accurate information retrieval thus become possible.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv:1301.3781*, 2013.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014, pp. 1532–1543.

[3] 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.

[4] 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.

[5] 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.

[6] S. Bartunov, D. Kondrashkin, A. Osokin, and D. P. Vetrov, "Breaking sticks and ambiguities with adaptive skip-gram," *CoRR*, vol. abs/1502.07257, 2015.

[7] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[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*, ser. SemEval '07, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 7–12.

[13] R. Mihalcea, T. A. Chklovski, and A. Kilgarriff, "The senseval-3 english lexical sample task," Association for Computational Linguistics, 2004.

[14] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[15] 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.

[16] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," in *HLT-NAACL*, 2013, pp. 746–751.

[17] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, pp. 137–186, 2003.

[18] L. White, R. Togneri, W. Liu, and M. Bennamoun, "How well sentence embeddings capture meaning," in *Proceedings of the 20th Australasian Document Computing Symposium*, ser. ADCS '15, Parramatta, NSW, Australia: ACM, 2015, 9:1–9:8.

[19] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[21] G. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*. Addison-Wesley Press, 1949.

[22] A. Kilgarriff, "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, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 103–111.

[23] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," 2014. arXiv: 1411.1607 [cs.MS].

[24] G. K. Zipf, "The meaning-frequency relationship of words," *The Journal of general psychology*, vol. 33, no. 2, pp. 251–256, 1945.

[25] R. I. Tengi, "Wordnet: An electronic lexical database, the mit press, cambridge, massachusetts," in, C. r. Fellbaum, Ed. 1998, ch. Design and implementation of the WordNet lexical database and searching software, p. 105.

[26] W. N. Francis and H. Kucera, "Brown corpus manual," *Brown University*, 1979.

[27] 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*, ser. SemEval '07, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 30–35.