CS224n Final Project

Trevor Standley
Student ID 05538335 SUNet: tstand
tstand@cs.stanford.edu

Abstract

I introduce a novel method for disambiguating word senses using a semi-supervised approach. I contrast this method with the current state-of-the-art approaches and show that my approach performs well and could potentially lead to fully unsupervised approaches with high accuracy.[1]

1 The Problem

In all or nearly all natural languages, there are words with several possible meanings or senses. These words are referred to as *polysemic*. For a polysemic word, the context within which an instance of the word appears disambiguates the intended sense for that instance of the word. For example, the following instances of the verb "missed" all have different senses:

- I missed the children.
- I missed my calling.
- I missed the target.
- I missed the period
- I missed the point.
- I missed the flight.

For many languages, polysemy is surprisingly common. For example, the most frequent 500 words in English have an average of over 23 distinct definitions in the Oxford English Dictionary. Yet, humans are typically capable of effortlessly determining the meanings of polysemes given context.

Multiple senses of a word can arise in many different ways. One way is the metaphorical use of a literal sense of a word. For example, "missed" is literally used to convey a situation in which the path of one entity does not intersect with the path or position of another entity, such as in "I missed the target." "I missed my calling" uses this literal sense in a metaphorical way through the life-is-a-journey metaphor, which linguists annotate as a distinct sense. Other examples of metaphorical senses of words include "She is a warm person" and "We had a deep discussion." Another way in which multiple senses of a word could arise is that words with originally different written forms and completely different meanings evolve to have the same written form, e.g., "bass," which could mean the lowest pitch range or a kind of fish.

Automatically determining the intended meaning of an ambiguous word is called word sense disambiguation (WSD), and is a core problem in natural language processing. WSD is necessary for many natural language processing tasks. For example, the meaning of the word "goal" in "He scored a goal" and "It was his goal in life" determines whether to use "gol" or "meta" for automatic translation into Spanish (Pal and Saha, 2015). Although many word sense disambiguation techniques have been shown to moderately improve performance for some tasks, none have been accurate, general, and fast enough to become common tools for NLP tasks.

2 Sense Labels

Much of the work on WSD aims to classify each instance of an ambiguous word into a WordNet synset (Fellbaum, 1998). Each WordNet synset represents a possibly singleton set of synonyms, and is annotated with a part of speech, a dictionary definition, and possibly links to hypernyms, hyponyms, and sister terms.

WordNet's senses have been criticised for being overly fine-grained to the point where even humans cannot distinguish between certain pairs

[1] This paper represents a work in progress.

of senses. For example, the WordNet entry for the word "bass" lists eight definitions for the noun form. Three of these definitions are for the fish sense, and five are for the musical sense. Some of the definitions are nearly identical, making it extremely difficult to automatically distinguish between them.

In order to mitigate this issue, some WSD systems use more coarse-grained senses, either by clustering senses in WordNet and choosing a canonical sense for each cluster (Navigli et al., 2007), or using OntoNotes senses, which were designed to achieve high inter-annotator agreement, ensuring that the sense distinctions are not overly subtle (Hovy et al., 2006). Unfortunately, OntoNotes does not attempt to create senses for parts of speech other than nouns and verbs.

Finally, some recent work on WSD uses BabelNet senses. BabelNet senses integrate WordNet senses and Wikipedia articles, and contain a mapping between words from multiple languages and its ontology. Unfortunately, BabelNet only contains nouns (which tend to be easier to disambiguate).

Because different papers report their accuracies with respect to different sense categories and corpora, it is usually not possible to compare accuracy measures directly.

3 Related Work

3.1 Early Research

Warren Weaver is credited for having first introduced word sense disambiguation as a computational problem in his 1949 memorandum on machine translation (Weaver, 1949), in which he discussed several impediments to machine translation and possible ways to overcome them.

Early techniques from the 1970s were rule-based and not very successful, and will not be summarized here. Later methods mainly fell into three categories: dictionary-based, supervised learning, and unsupervised or semi-supervised learning.

3.2 Dictionary Methods

In 1986, Michael E. Lesk introduced a dictionary-based approach that counts the number of overlapping words between the context of an instance of a polyseme and each of the polyseme's dictionary definitions, choosing the definition with the greatest number of overlapping words (Lesk, 1986). Recent analysis of Lesk's algorithm places its accuracy at about 42%. Extensions of Lesk's algorithm that use all relevant text from WordNet, lemmatization, and smart back-off to the most common sense are able to achieve 58%[2] accuracy (Vasilescu and Lapalme, 2004). These so-called dictionary-based methods are convenient because they allow any word in the dictionary to be disambiguated, unlike supervised learning approaches that require a tagged corpus of contexts for every sense of every ambiguous word.

3.3 Supervised Learning Methods

The most accurate word sense disambiguation systems use supervised learning. Although these systems are accurate, they are impractical for most applications because they can only disambiguate polysemes that have sense-labeled context corpora.

To the best of my knowledge, the state-of-the-art system for supervised WSD trains a naïve-Bayes classifier on top of latent Dirichlet allocation (LDA) topic features (Cai et al., 2005), narrowly beating the competition in the supervised section of SemEval-2007 with an accuracy of 88.7%[3].

3.4 Unsupervised and Semi-Supervised Learning Methods

More recent work has focused on unsupervised leanring methods because they do not require training on labeled data for every polyseme.

From the SemEval-2007 results, the state-of-the-art semisupervised learning method for WSD was (Chan et al., 2007). The authors make use of Chinese-English parallel corpora which they align for creating pseudo-labeled data upon which the authors train an SVM model. The authors method requires that every English polyseme to be disambiguated be manually tagged with one or more appropriate Chinese translations. The authors manually entered such translations for the top 730 nouns, the top 326 adjectives, and the top 190 verbs. The system falls back to the most frequent sense baseline for cases in which they do not

[2]The accuracies reported are highly dependent on the particular evaluation rules. The numbers reported for (Vasilescu and Lapalme, 2004) are from the SENSEVAL2 English All Words task. For reference, the best system for that track achieved 69% accuracy.

[3]SemEval-2007 task 17, subtask 1 used OntoNotes senses, for which the baseline using the most frequent sense was 78.0%.

have sense translations. Their approach achieves an accuracy of 82.5% on the course grained word senses[4]. Though their performance was not substantially above the baseline, this technique for generating training data has become a standard trick in many more recent works.

The authors of (Trask et al., 2015) show that the performance of dependency parsers can be improved using sense specific word vectors. They show that their disambiguated vectors have the highest cosign similarity with synonyms of the same sense. Unfortunately, the authors train their vectors only after disambiguating their corpora and the methods used to disambiguate the words are weak. The authors mention part-of-speech tagging each word and using the POS to disambiguate, however this would be unable to distinguish between senses of the same POS. Alternatively they mention using the inferred sentiment of the document to disambiguate between certain senses of certain words, but most senses of most words have neither positive nor negative sentimate.

3.5 Progress

Much of what we know about the performance of WSD systems comes from the various SemEval and Senseval contests that have been run over the years. Unfortunately, the setups change dramatically with every new contest, making it difficult to track the progress of WSD systems. The cordinators of the SemEval-2007 Coarse-Grained English All-Words task note that at the least the improvement over the most frequent sense (MFS) baseline can be tracked (Navigli et al., 2007). Unfortunately the baseline-to-winner delta did not improve between 2004 and 2007, and actually decreased between 2007 and 2013. One explanation could be waning interest in the problem. In 2013 only six contest entrants representing three teams competed. Another possibility is that the general trend towards courser senses and fewer parts of speech (in 2010 and 2013, only nouns were considered) lead to increased baseline performance, but not increased winner scores.

It is clear, however, that progress has not dramatically improved in the past decade despite the advent of big data, deep machine learning, and continued work on linguistic resources such as WordNet.

[4]For comparison, the WordNet first sense baseline score was 78.9%.

I *misplaced* my glasses.	I *misplaced* the game.	I *misplaced* their trust.
I *gained* my glasses.	I *gained* the game.	**I *gained* their trust.**
I *won* my glasses.	**I *won* the game.**	I *won* their trust.

Table 1: Using substitute words to disambiguate the verb 'lost' in three different contexts. Bold indicates the most likely semantic for each context (column).

4 My Approach

I approach this problem with the following insight: the semantics of certain substitute words seem likely in the context of some senses of a polyseme, but unlikely in other senses. For example, we see that the word 'won', makes semantic sense in the context of 'lost' in 'I lost the game', because you can win a game, but the semantics of replacing 'lost' with 'won' don't seem very likely in the context of 'I lost my glasses' because glasses aren't typically the type of thing you win.

In Table 1 I present a more complete example. Here we try to discern the sense of the word 'lost' with respect to the three different contexts, 'I lost my glasses', 'I lost the game', and 'I lost their trust'. We see that we can use the substitute words 'misplaced', 'won', and 'gained' to classify each usage of the word 'lost' into a different sense category, thereby disambiguating the meaning of the word given the context.

In order to run this procedure, we need a way of telling how likely a given word is in a certain context, and a way of finding the most representative word or words for each sense.

4.1 Contrastive Estimation

We are in need of a computable function $f(word, context) \in [0, 1]$ for determining how well a word semantically works in a context.

We train our function $f()$ using a special case of Contrastive Estimation (CE). (Smith and Eisner, 2005). This process requires no labeled data, but can be trained on a large corpus of natural language text. We generate training examples of the form $(word, context, label \in \{true, false\})$ from this corpus in the following way. For each sentence in the corpus, we generate a single example. First, we determine if the example should be positive or negative by flipping a coin. If we decide the example should be positive, we pick a random word to be the $word$, and use the

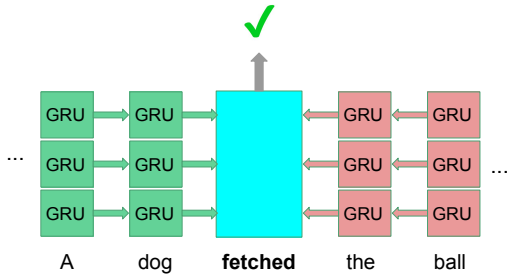

Figure 1: Our recurrent neural network architecture. The left and right recurrent structures are stacked GRU nodes (Chung et al., 2015) with 256 hidden units each. The middle node is made up of densely connected rectified linear units with two hidden layers. The output is a sigmoid unit. The model has only 2.3 million parameters total.

rest of the sentence as $context$. If we decide to generate a negative example, we choose a random word to replace, and our training example becomes $(random\_word, context, false)$. Using the Gigaword corpus of newswire text (Parker et al., 2011), we can generate unique examples until the training converges (i.e. we never have to re-use training examples).

At both training and test time, I restrict the context to at most 30 words on either side of the $word$.

We use a recurrent neural network as our function approximator. The architecture is described by Figure 1. Training takes about 24 hours to converge and achieves a 87.5% accuracy in determining whether the example word was replaced. 100-dimensional pretrained GloVe vectors (Pennington et al., 2014) are used as our embedding layer. We do not propagate gradients back to the embedding layer.

4.2 The Substitute Words

For the example in Table 1, we use the substitute words 'misplaced', 'gained', and 'won' to disambiguate between the three senses for 'lost'. 'misplaced' is a synonym and 'gained' and 'won' are antonyms, however, under this framework, other word relations could work just as well. For example when trying to disambiguate between the musical form and the fish form of 'bass', we could use a different kind of fish, such as 'trout', and a different but related musical term such as 'tenor'.

By choosing a single word for each sense, I would be limiting my accuracy. Instead, my system chooses several words for each sense and averages their resulting probabilities for each context. Concretely, instead of determining whether $f(trout, context) > f(tenor, context)$ to disambiguate a use of 'bass', I can check if $.25 * f(trout, context) + .25 * f(salmon, context) + .25 * f(tuna, context) + .25 * f() > .5 * f(tenor, context) + .5 * f(soprano, context)$, which leads to superior results.

The challenge becomes picking the right words for each sense. I can think of four categories of methods to go about this. First, we can hand pick the words. Second, we could harvest the words from a dictionary or database like WordNet. Third, we could use unlabeled clustering. And finally, we can do supervised training on labeled data.

My current work focuses on the last option, learning from labeled data, but in the future, I think the second option is most promising, possibly augmented with labeled data and supervised clustering.

4.3 Supervised Training

In order to find the best set of words for each sense, I simply score each word in a list of words, and choose the best n words by score. The score for each word for each sense is simply the average $f()$ value for that word over all the contexts labeled for that sense, less the average $f()$ value for that word over all the contexts for other senses.

More concretely, given a set of training examples t for a polyseme of the form (s, c), where s is a sense, and c is a context, the score for a word w is given by the Equation 1.

$$score(s, w) = mean(\{f(w, c) : (s, c) \in t\}) - mean(\{f(w, c) : (s', c) \in t \wedge s' \neq s\}) \quad (1)$$

Choosing the top 30 words from a list of the 8000 most used English words with the same POS as the polyseme seems to work well, although this parameter hasn't been tuned.

5 Tricks

On its own, this method performs well enough to beat the baseline in our experiments, but not well enough to be compelling. I use two main tricks to improve performance drastically. The first is a method for taking into account how common a sense is for a particular word. The second is preventing the network from learning grammar.

5.1 Sense Frequency

The most effective trick for improving performance is taking into account the natural frequency of each sense. The quality of the most-frequent-sense baseline (MFS) is surprisingly high, and many systems use MFS as a fallback when the system doesn't have enough information. Instead of using the sense frequencies as a fallback, my system assigns a score to each sense at test time and chooses the sense with the best score. The score as described above is simply average $f()$ score of the top n words chosen for that sense. However, augmenting the score s with the frequency F of the sense from the training data helps tremendously.

Concretely, the augmented score $s' = s^\lambda F$ performs much better than the unaugmented score. The parameter λ was chosen to maximize the score on the *training* set.

Choosing the sense that maximizes the augmented score improves the system's accuracy by 9.2 percentage points and is therefore invaluable.

5.2 Preventing the Model From Learning Grammar

Another trick I use to improve performance is how I generate the CE data. First, I use the Stanford POS tagger (Toutanova et al., 2003) to tag each word in a training example. Next I lemmatize the randomly chosen *pivot* word. Finally, if the example is to be a negative example, I make sure to replace it with the lemma of a random word with the same part of speech.

Without these changes, the network can learn grammar instead of semantics in order to tell if the pivot word is an impostor. Early experiments showed a 2% improvement in system accuracy as a result of this change.

6 Experimental Results

Experiments were run using the Senseval-3 English Lexical Sample task fine-grained framework. My solver achieves an accuracy of 70.9% compared to a most frequent sense baseline score of 55.2% and a winning score of 72.3%[5]. Though this contest is not the most recent one on WSD, it seems that not much progress has been made by winning solvers in subsequent years. Therefore, I suspect that my solver's accuracy represents nearly state-of-the-art performance. Furthermore, most of my parameters were not carefully tuned, so higher performance can probably be achieved after tuning the parameters.

[5]Note that the winning entry used a voting scheme from an ensemble of independently developed classifiers.

7 Implementation

The code for the project is written in Python. I use Keras (Chollet, 2015) to do the neural network training. The implementation is in several scripts. The first script preprocesses the unlabeled corpus, adding POS tags. The next script learns the function $f()$ with contrastive estimation. The next script reads the Senseval-3 contest files, finds the appropriate replacement words, and classifies the test set. Finally the last script post processes the sense weights, taking into account the sense frequency. The last script reports the score, and outputs a file for the Senseval-3 scoring code to analyze (though the scores reported are the same).

Finally, to ease testing, I create a wrapper class that loads the Keras models and keeps them in memory. This way I don't have to re-compile the model to test subsequent changes to the third stage.

8 Future Work

The most promising direction for taking this work is to the fully unsupervised realm. Currently, the only supervised process is in the finding substitute words step. It seems that the next thing I should try should be automatically choosing suitable words from WordNet.

Unsupervised context clustering is also a promising direction. By clustering contexts by which words work best in them, the proper senses could be induced without a dictionary. Better yet, modifying the M step for EM clustering so that unlabeled contexts are more likely to cluster into WordNet categories with respect to the above method could provide not only a mapping into canonical WordNet senses, but also serve to approximately label data for training.

It's also important to compare to non-neural CE baselines and to test my framework on other WSD contest data.

Finally, the network architecture I chose is problematic for a number of reasons. First it requires relevant long term context to propagate through a long recurrent chain, degrading its impact. A better architecture might only use a small window of context, and then summarize the rest of the context words in some other way (perhaps con-

volutional max pooling, or more simply, just a normalized average of their GloVe vectors). Second, I think the center node is too deep, slowing training. Finally, the network doesn't contain many parameters, but still trains slowly. I think the architecture could be improved drastically, leading to better performance and faster learning.

9 Conclusion

This paper describes the general problem of word sense disambiguation and previous attempts to solve it. A novel method for solving WSD is introduced, based on the combination of contrastive estimation and word substitution. This method achieves nearly state-of-the-art performance and is promising in its simplicity (only a single model), novelty, ability to combine with other methods, and potential to lead to fully unsupervised WSD.

References

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. Nus-ml:improving word sense disambiguation using topic features. In *In Proceedings of SemEval-2007. Association for Computational Linguistics, 2007*, pages 524–531. IEEE Computer Society, 2005.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. Nus-pt: Exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 253–256, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1621474.1621528.

Franois Chollet. Keras. https://github.com/fchollet/keras, 2015.

Junyoung Chung, Çalar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. 2015.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: The 90 In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1614049.1614064.

Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM. ISBN 0-89791-224-1. doi: 10.1145/318723.318728. URL http://doi.acm.org/10.1145/318723.318728.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 30–35, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1621474.1621480.

Alok Ranjan Pal and Diganta Saha. Word sense disambiguation: a survey. *CoRR*, abs/1508.01346, 2015. URL http://arxiv.org/abs/1508.01346.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition. *Linguistic Data Consortium*, 2011.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 354–362, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219884. URL http://dx.doi.org/10.3115/1219840.1219884.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter*

of the Association for Computational Linguistics on Human Language Technology - Volume 1, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073478. URL http://dx.doi.org/10.3115/1073445.1073478.

Andrew Trask, Phil Michalak, and John Liu. sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *ICLR under review*, abs/1511.06388, 2015. URL http://arxiv.org/abs/1511.06388.

Florentina Vasilescu and Guy Lapalme. Evaluating variants of the lesk approach for disambiguating words. In *In LREC*, 2004.

Waren Weaver. Translation. *Reproduced by permission of the Rockefeller Foundation Archives*, 1949. URL http://www.mt-archive.info/Weaver-1949.pdf.