CS224d Final Project

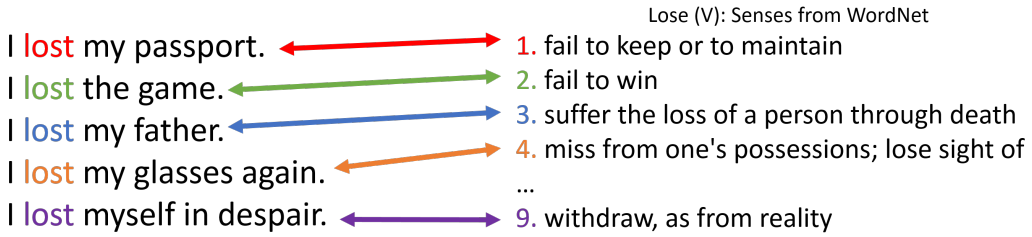
Trevor Standley
Student ID 05538335 SUNet: tstand
tstand@cs.stanford.edu

Abstract

I present two methods for disambiguating word senses. The first is completely unsupervised, and relies on WordNet information to disambiguate senses, The second uses a semi-supervised approach. I contrast my methods with state-of-the-art methods and speculate that a combination of techniques might be effective. Finally, I experiment with a technique that can leverage large quantities of supervised data during the deep-learning phase, which was normally unsupervised.[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 "lost" all have different senses:



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, "lost" is literally used to convey a situation in which one can no longer locate an object, such as in "I lost my glasses." "I lost my father" uses this word in a metaphorical way. 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

[1] This paper represents a work in progress.

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 of senses. For example, the WordNet entry for the word "bass" lists eight definitions for the noun form. Three of these definitions are for a fish, and five have to do with music. 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, corpora, and even sense category versions, 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.

[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.

3.3 Lexical Sample Systems

The most accurate word sense disambiguation systems use supervised learning. Lexical Sample systems aim only to be able to disambiguate only a small set of words (for which ample training data has been obtained). Although these systems are accurate, they are impractical for most applications because they can only disambiguate polysemes for which ample training data exists.

The state-of-the-art system for Lexical Sample 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 All-Words Systems

In contrast, all-words systems aim to disambiguate all encountered content words (nouns, verbs, adjectives, and adverbs). All of the competitive all-words systems are supervised, typically using a dataset like SemCor described below.

It Makes Sense is the state-of-the-art all-words system for WSD (Zhong and Ng, 2010). 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. This methiod, which was inspired by a previous work (Chan et al., 2007), requires that every English polyseme to be disambiguated is manually tagged with one or more appropriate Chinese translations. The authors manually entered such translations for thousands of the top nouns, verbs, and adjectives. The system falls back to the most frequent sense baseline for cases in which they do not have sense translations. Their approach achieves an accuracy of 82.6% on the course grained word senses of SemEval-2007. For comparison, the WordNet first sense baseline score was 78.9%. They also achieve roughly state-of-the-art performance for the SensEval-2 and SensEval-3 competitions. The fact that the state-of-the-art-system performs so close to baseline has meant that few systems can make use of WSD systems for downstream tasks.

3.5 SemCor

In order to statistically determine the frequency order of WordNet definitions, the creators of WordNet underwent an effort to tag 35k sentences of text from the brown corpus. The result was SemCor (Shari Landes and Fellbaum, 1998). This sense tagged corpus contains disambiguated sense tags for more than 200k content words. It is an invaluable resource for anyone working on WSD.

Unfortunately, the words were tagged uniformly across the corpus. Words like forms of the verb 'to be' are tagged tens of thousands of times while many less common words are missing completely. Due to the statistics of word senses less than 30% of senses encountered in test sets have more than 5 examples in SemCor.

3.6 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.

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

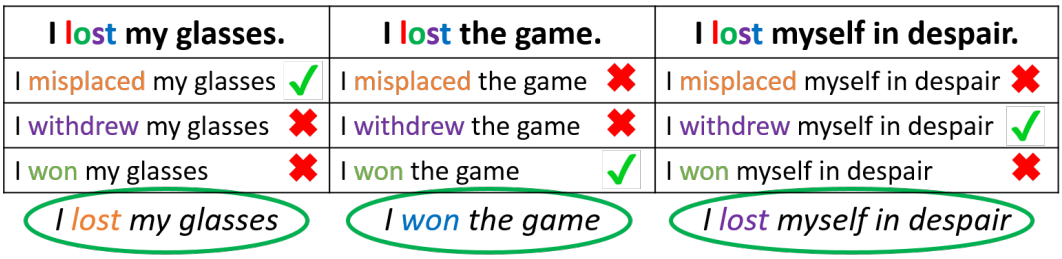


Figure 1: Using substitute words to disambiguate the verb 'lost' in three different contexts.

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 myself in despair'. We see that we can use the substitute words 'misplaced', 'won', and 'withdrew' 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 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 English Gigaword 5 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 4 words on either side of the $word$, however the model is also presented with a vector that is the average of all context vectors as input. Because of this context vector, increasing the window size beyond 4 on either side has no measurable affect on performance.

We use a recurrent neural network as our function approximator. The architecture is described by Figure 2. Training takes about 24 hours to converge on a single Nvidia GTX 970 and achieves a 88.6% accuracy in determining whether the example word was replaced. 300-dimensional pretrained GloVe vectors (Pennington et al., 2014) are used to initialize our embedding layer. The embedding layer contained the top 100k words in our vocabulary.

Eleven different architecture combinations were tried. None performed statistically significantly better than the one pictured in Figure 2. An alternate, but far more complicated architecture had fewer parameters and reduced iterations to convergence, but did not produce higher accuracy.

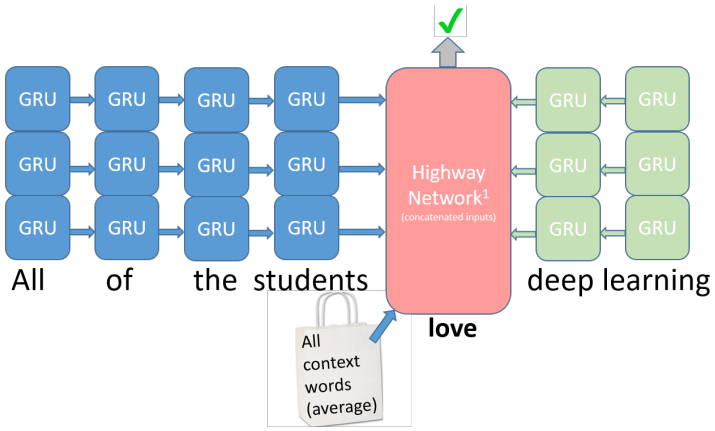


Figure 2: Our recurrent neural network architecture. The left and right recurrent structures are stacked GRU nodes (Chung et al., 2015) with 300 hidden units each. Our word embedding layer contained 100k 300 dimensional word vectors. The middle node is made up of a single highway network layer (Srivastava et al., 2015) that takes the context from the right, the context from the left, the BOW average context vector, and the word in question. The output is a sigmoid unit representing the probability that the word was not changed. The model has only 4.3 million parameters total.

4.2 The Substitute Words

For the example in Table 1, we use the substitute words 'misplaced', 'withdrew', and 'won' to disambiguate between the three senses for 'lost'. 'misplaced' is a synonym and 'won' is an antonym, 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 $w_1 * f(trout, context) + w_2 * f(salmon, context) + w_3 * f(tuna, context) > w_4 * f(tenor, context) + w_5 * f(soprano, context)$, which leads to superior results.

The challenge becomes picking the right words for each sense.

We use several methods for picking the words depending on our training regime.

For words with scarce data, we simply obtain synonyms, antonyms, and sister words from WordNet, and we weight them evenly.

For words with enough labeled data for each sense, we try the top 5000 words of the appropriate part of speech, and we treat their activations as features. We then use the default scikit-learn feature selection (χ^2 method) against the appropriate sense to pick the best 300 replacement words, and we train a multi-class SVM to obtain the weights. In order to mitigate overfitting of the SVM, a held out set of examples from each word is used to tune a regularization parameter.

4.3 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.

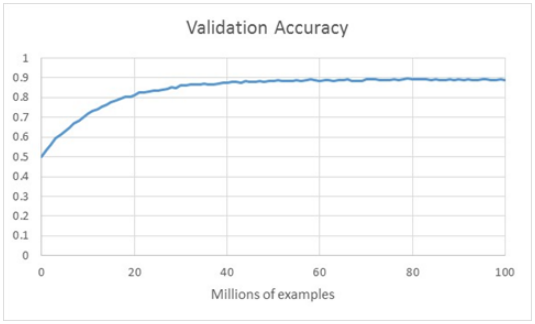


Figure 3: Training accuracy over time.

5 Training

As mentioned, we were able to generate effectively infinite contrastive estimation training examples from the Gigaword 5 corpus.

To better see how the examples were created, consider the following sentence as if it were in the unlabeled corpus:

`All of the students love deep learning.`

The first step is to pick a 'pivot' word randomly from the sentence's content words. In this case, we chose 'love'. Next we choose whether to create a positive or a negative example out of this sentence using a fair coin flip. If we choose to create a positive example, the training example is:

`All of the students` **`love`** `deep learning.`

If we choose to create a negative example, we then need to choose a random replacement word with the same part of speech. It is also extremely important that the random word be chosen in proportion to the words natural frequency in the training set, otherwise, the network will learn to give low scores to infrequent words, and the accuracy will be artificially high (it'll also take longer to train). Lets assume we chose the word **jump**. We then have the negative example:

`All of the students` **`jump`** `deep learning.`

Because we never need to use the same instance twice training accuracy is the same as validation accuracy if training accuracy is measured before an update (though we did track both). Before one hundred million examples, the accuracy plateaus.

We used the Adam optimizer in Keras (Chollet, 2015) with a learning rate of .0005. I used a batch size of 512.

5.1 WSD Supervised Contrastive Estimation

Given enough training data, our contrastive estimation model could be trained to theoretically produce better results for WSD. The main difference would be how the data is generated.

For every labeled example, we can generate a set of examples for our model. For positive examples, we replace the pivot word ('love' in this case) with a replacement word of the correct synset (such as 'like', 'enjoy', or 'hate'). For negative examples, we replace the pivot word with a replacement word of the incorrect synset (such as 'admire' or 'copulate'). This will cause the model to give higher scores to correct replacement words than it gives for incorrect replacement words.

We experimented with this type of training using SemCor. First we held out a set of 20k instances for validation. We were able to improve the contrastive estimation accuracy on these 20k validation instances from 65% to 73%, but this gain did not translate to a gain on our full WSD system for any of the contest results. I believe this method is still promising, especially with larger datasets such as the one obtained for It Makes Sense using parallel corpora.

	SensEval-2	SensEval-3
SOTA System	69.0%	67.6%
My System	66.8%	65.3%
Baseline	61.9%	62.4%

Figure 4: Results on SemEval-2 and SemEval-3.

5.2 SVM Training

As mentioned, when SemCor contains at least ten examples for at least two senses of a word[4], I train an SVM multiclass classifier with which to disambiguate that word. Given the small amount of data, and my choice of 300 top replacement word features, the SVM can easily over-fit the data. To mitigate this problem, I employ strong L2 regularization. For each word, I hold out 20% of the examples during cross-validation to choose the regularization parameter. After the parameter is chosen, I roll back in the validation data before training the final model.

6 Experimental Results

Figure 4 shows my results on the SensEval-2 and SensEval-3 contest datasets, and is the percent of the time that exactly the correct sense was chosen. For each of these datasets the inter-annotator agreement was in the 70% range. Our results fall in between the baseline and the SOTA system (which was It Makes Sense for SensEval-3). Although our solver often deviates from choosing the most frequent sense in many examples, the accuracy is not wildly different from this baseline. This suggests that when the model deviates from the top sense it is wrong almost as often as it is right, but it also suggests that the sense chosen is more variable, and could therefore be useful for example by training per-sense word vectors.

7 Conclusion and Future Work

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.

Although my results do not surpass the state of the art yet, I believe they can be substantially improved. First, my results shown do not use features from the current state-of-the-art. Ideally I would build my features into the open source It Makes Sense solver which is state-of-the-art as it is. Second, I have discovered a much larger labeled dataset (1 million instances) that was extracted from parallel corpora which can be used to improve the SVM training as well as help me further explore the WSD supervised contrastive estimation idea.

Ultimately, the replacement word concept has proven useful, though it is not yet a wholesale replacement for traditional methods.

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.

[4]these parameters have not been optimized.

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.

Claudia Leacock Shari Landes and Christiane Fellbaum. Building semantic concordances. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 8, pages 199–216. Bradford Books, 1998.

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.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *CoRR*, abs/1507.06228, 2015. URL http://arxiv.org/abs/1507.06228.

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.

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.

Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858933.1858947.