Word Sense Tagging Over Continuous Space with Supersense Back-off

Boyu Zhang
Advisor: Professor Aaron Steven White
University of Rochester
Spring 2019

Abstract

In this paper, we propose a novel transfer learning approach utilizing distributional embeddings of words to solve the word sense disambiguation (WSD) task. Our model is different from previous statistical WSD models: we jointly optimized the word sense predictions and the word sense definitions in a high-dimensional continuous vector space (sense space) instead of discrete probability distributions. We also managed to integrate deterministic symbolic knowledge of supersenses from the WordNet (Miller, 1995) to our statistical model. Our methods have been shown very effective in multi-label word sense tagging for both known words and unknown words.

Background

Many words have multiple senses. Ambiguous words like “bank” – which has both “financial institution” and “side of a river” senses – are a typical case of this. Humans can distinguish between the following sentences easily:

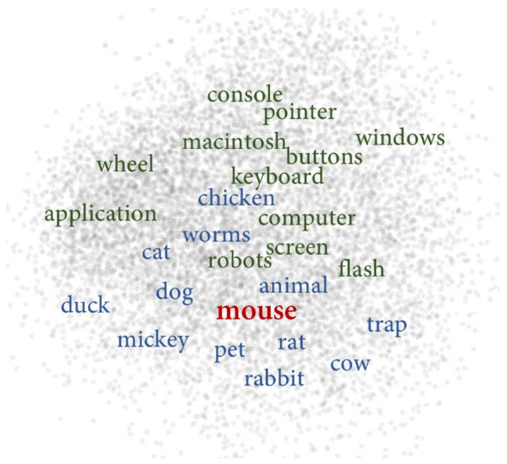
1. “They pulled the canoe up on the bank.”
2. “He cashed a check at the bank.”

But for machines, this is an unsolved problem. Word sense disambiguation (WSD) plays a vital role in the field of natural language processing and computational linguistics.

In this paper, our transfer learning WSD model is based on distributional embeddings of words. It is worth noticing that pre-trained probabilistic language models can be contextual or non-contextual. In the case of non-contextual (e.g., word2vec), one particular word will only have one embedding despite the context of the word; in the case of contextual embeddings, however, the same word will have different embeddings in different sentences which capture the unique contextual information. Obviously, WSD can only be achieved by analyzing the context of the ambiguous word. Thus, it is crucial to use contextual embeddings in this transfer learning model.

In this paper, we are using the Embedded Language Model (ELMo) developed by the Allen Institute of AI (Peters et al., 2018). Yes, its name is not very impressive. It is a contextual language model trained on 1B Word Benchmark (Chelba et al., 2013) corpus. Given a sentence, ELMo produces contextual word embeddings for all words in this sentence. These embeddings are interrelated and thus are called contextual embeddings: the same word in different sentences will have different ELMo embeddings.
As readers may notice, non-contextual embedding is a one-vector-per-word model, while contextual embedding is similar to a one-vector-per-sense model. There is a long-lasting

problem in non-contextual models called "Meaning Conflation Deficiency" (Camacho-Collados et. al., 2018).



As illustrated in the above figure, in non-contextual models, the word "chicken" is close to the word "computer" due to the ambiguity of the word "mouse" in word2vec. Conversely, though the contextual embedding is not strictly identical for the same sense of an ambiguous word in different sentences, it is sensitive to the word context and deepens the distributional representations to the sense level instead of merely the word level. Solving the WSD task on sense level is one of the most important ideas throughout this paper.

Previous Work

Mooney et. al., 2010 made the first try on sense-level WSD by computing multiple distributed embeddings for one word based on its contexts from corpus data. Specifically, the researches clustered all the contexts of the target word in the corpus and calculated the mean of each of the clusters as sense embeddings. This is often referred to as “multi-prototype” (Mooney et. al., 2010).

Later works followed the idea of “multi-prototype” mostly and treated the WSD task as a supervised classification problem. Various models were able to output the probability distribution over all possible senses of the target word using machine learning methods including sequence-to-sequence learning (Raganato et. al., 2017), attention mechanism (Tang et. al., 2018), and neural n-gram structure (Neelakantan et al., 2015).

Dataset Statistics

In this paper, we are using the Universal Decompositional Semantics on Universal Dependencies (White et. al., 2016). It contains annotations for noun word senses based on the senses listed in the WordNet. The data format is shown in the image below:

```
******************* Data Example **********************
Sentence: ['on', 'August', '9', ',', '2004', ',', 'it', 'be', 'announce', 'that', 'in', 'the', 'sp
ring', 'of', '2001', ',', 'a', 'man', 'name', 'El', '-', 'Shukrijumah', ',', 'also', 'know', 'as',
'Jafar', 'the', 'Pilot', ',', 'who', 'be', 'part', 'of', 'a', '"', 'second', 'wave', ',', '"', 'ha
ve', 'be', 'case', 'New', 'York', 'City', 'helicopter', '.']
Annotator Response, i.e., true label: [1, 0, 0, 0, 0, 0]
Target Word Index: 12
All senses for the target word: ['spring.n.01', 'spring.n.02', 'spring.n.03', 'spring.n.04', 'giv
e.n.01', 'leap.n.01']
All definitions (in order of its senses from WordNet): [['the', 'season', 'of', 'growth'], ['a',
'metal', 'elastic', 'device', 'that', 'returns', 'to', 'its', 'shape', 'or', 'position', 'when',
'pushed', 'or', 'pulled', 'or', 'pressed'], ['a', 'natural', 'flow', 'of', 'ground', 'water'],
['a', 'point', 'at', 'which', 'water', 'issues', 'forth'], ['the', 'elasticity', 'of', 'somethin
g', 'that', 'can', 'be', 'stretched', 'and', 'returns', 'to', 'its', 'original', 'length'], ['a',
'light,', 'self-propelled', 'movement', 'upwards', 'or', 'forwards']]
******************************************************
```

The sample sentence and index of the target word are given in the first place. Punctuations were preserved since they contain important effects when generating contextual information. In the annotator response vector, "1" at index "i" means the ith sense of the target word in the WordNet is the correct sense in the given sentence, and vice versa for "0". Each of the senses also has a serial number, e.g., "spring.n.01". For the above example, the annotator marked the first sense as 1 (True), and the corresponding WordNet definition is "the season of growth". The other senses are 0 and not suitable for this sentence. Annotators are allowed to check multiple correct senses.

There are 105274 sentences (72314 training samples, 16400 development samples, and 16560 test samples) with target words extracted from the Universal Dependency of English (UDE) dataset by the Stanford University (Universaldependencies.org, 2019). For each target word, it has 3.7 different senses on average across the dataset, given the definitions from the WordNet. The vocabulary size (distinct words) of the first 15000 training data is 3129.

Model Architecture

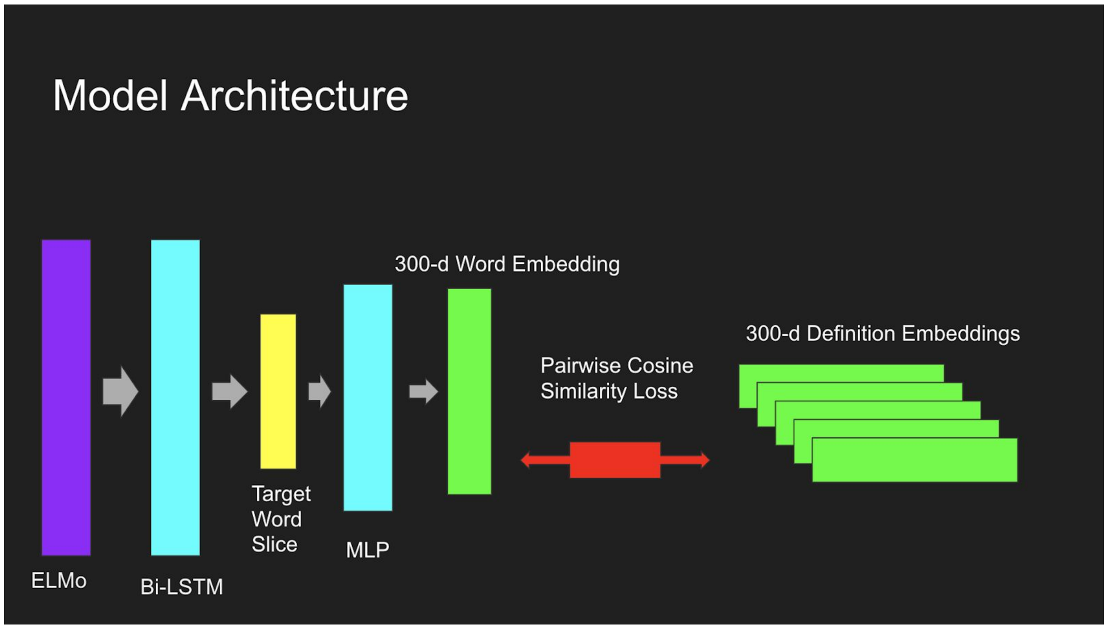
Though the aforementioned works obtained decent results on the WSD task, these supervised classifiers are not flawless. Different ambiguous words have different numbers of senses. Thus, a universal WSD model must either output the probability distribution over all senses of all words in the vocabulary or tailor a classifier for each word in the corpus. Considering the vocabulary provided by the WordNet (Miller, 1995), the former requires a softmax calculation of size up to three hundred thousand. Normalizing the discrete probability distribution of such large size by softmax is computationally inefficient. It is also not reasonable to pad the probability output such that the word “bank” has a potential sense choice of “spring”. The latter suffers from the problem that there is hardly enough training data for each individual word. For example, for the single word “spring”, one needs at least hundreds of samples with “spring” as the target word to train the tailored classifier.

Here we introduce an alternative idea: sense embedding. This is not a new idea (Camacho-Collados et. al., 2018; Kumar and Tsvetkov, 2019), but it will avoid the above problems in classification. The fixed-size sense embedding is designed to represent the accurate sense of the target word in the given sentence in high-dimension space. More importantly, such sense embedding lays in the same vector space -- the sense space -- with all the possible definitions of the target word. We modified the output layer to be a universal layer with an output size [300, 1], treating this as the sense embedding but do not apply any activation for classification. Thus the values in the tensor are not bounded between 1 and 0 like the probability but have the potential to explore the whole vector space. After that, for a example word with an annotator response [1, 0, 0], we randomly initialize 3 vectors for each element in this response, treating them as the sense vector of definitions, laying in the same vector space as the output tensor of our model. In the rest of this report, I will refer the sense vector of definitions as definition embedding and the output tensor as predicted embedding.In a nutshell, the sense embedding is the projection of the sense of the target word in high-dimensional sense space, and all possible ground truth definitions of the target word also have their corresponding embeddings in the same vector space.

Our distributional model utilizes sense embeddings for WSD, and it is not a classifier operating over discrete probability distribution. Before proceeding to details of the model, there are two naming conventions we used throughout the rest of this paper:

1. The predicted sense embedding for word w: sense embedding.
2. The ground truth embeddings of senses s1, s2, s3, …, sn for the word w given by the WordNet: definition embeddings.

The model architecture is shown in the figure below:



First, for a target word w with n senses, we can generate its sense embedding by the following steps (the intermediate word embedding size is 256; the sentences are all padded to have the same length with the longest sentence):

1. Passing the given sentence from UDE to ELMo; getting the contextual embedding of all words in the sentence (input size: max length of the sentence; output size: max length of the sentence * word embedding size). There are 3 layers of Bi-LSTMs (Schmidhuber, et.

al., 1997) and 23040 weights in ELMo, and they are freezed for this transfer learning process.

2. Passing the ELMo sentence embeddings through a new Bi-LSTM; using the Bi-LSTM to capture rich semantics of the particular word sense in the given sentence sequence by recurrent backpropagation through time step (input size = output size = max length of the sentence * word embedding size). There are 2 layers and 2*3840 weights in the bidirectional LSTM.
3. Slicing out the target word embedding at the given index (input size: max length of the sentence * word embedding size; output size: word embedding size).
4. Passing the intermediate word embedding through a multi-layer perceptron with one hidden layer (input size: word embedding size; hidden layer size: 512; output size: sense embedding size). There are 256*512+512*300 weights in this fine-tuning MLP. This MLP will project the word embedding to the 300-d sense space as the sense embedding.

Second, we randomly initialize n definition embeddings (n is given by WordNet) for the target word w. Thus, for each word, we not only have the sense embedding produced by the transfer learning model but also have n definition embeddings as its ground truth.

The loss is defined as cosine similarity loss, and it is propagated bi-directionally to both the forward model and the definition embeddings. The optimization process is discussed in detail in the next section.

Optimization

So far, we have a forward model that produces a 300-d sense embedding (SE) and n random definitions embeddings (DE1...DEn) for the given target word laying in the same sense space. During the training process, we iterate through all definitions embeddings for the target word w:

1. If the DEn is the correct sense for w given the annotator response (1 in the response vector), pull them closer (increase similarity).
2. If DEn is the wrong sense for w given the annotator response (0 in the response vector), push them away from each other (decrease similarity).
3. Loss is calculated by cosine similarities (dissimilarities). It is then propagated back (performing gradient update) not only to the forward model but also to the definition embeddings. In the case of dissimilarity, we simply negate the loss.

Notice that “pulling” and “pushing” are represented by changes in cosine similarities between the sense embedding and the definition embeddings.

Thus, we are jointly optimizing:

1. The ability of the model to project word sense to the sense space accurately (generating the sense embedding).
2. The ability of the model to locate definition embeddings in the sense space accurately (optimizing definition embeddings).

This is a modified version of the Expectation-Maximization algorithm. The metaphor here is that the first step is assigning a word to a cluster, and the second step is correcting the mean of the clusters. Notice that, on one hand, the loss backpropagation for the sense embedding (forward model) will be accumulated across all its definition embeddings. On the other hand, for each definition embedding, we only update it w.r.t. its own loss comparing to the sense embedding. This means that one particular definition contributes only a small amount of correction normalized by the size of all definitions.

After the training process, we are expected to have a model that can generate accurate word sense embeddings and a bunch of definition embeddings that lay with certain patterns (contextual similarity, unit-type polysemy, etc.) in the vector space. The exact location of the definition vectors may vary since they are randomly initialized, but the important properties are the relative positions of definitions embeddings to the predicted sense embeddings and between definitions embeddings themselves. These relative positions imply semantic relationships between different senses.

For a given word, we can test the performance of the model by:

1. Generating the predicted sense embedding.
2. Masking out the annotator responses.
3. Comparing the similarities between the sense embeddings and all its definition embeddings.
4. Picking the most similar one.
5. Checking if the pick has an annotator response of 1.

As you may notice, this model only works for words within the lexicon of the WordNet. Otherwise, we do not have the definition embeddings.

Supersense

After the joint optimization of the model, we seek a solution to the problem of unknown word aforementioned. Performing WSD on unknown words may be hard to understand at first, but we are introducing a low-level WSD on unknown words here. We introduce an idea called "supersense" (Ciaramita, 2003). Supersense, or Lexicographer, is a universal ontology across all vocabularies. It acts as a "generalization" and "abstraction" of word senses and is not affected by unseen words. WordNet generalizes the abstract "types" of all nouns into 26 supersenses: time, object, etc. Each sense of a noun belongs to a certain supersense. For example, “spring” the season belongs to the supersense “time”.

We initialize 26 300-d vectors for all supersense as supersense embeddings. Each supersense embedding is initialized as the mean of all its children (provided by the WordNet). During the optimization of sense and definition embeddings, we also optimize the embeddings of supersenses in the same way: increase the similarity towards the correct supersense and vice versa. Besides, we always increase the similarity between the child definition embedding and the parent supersense after each update in a way that the supersense remains as the mean of its children.

Here is a native example: suppose the model knows "apple", "banana", and "orange", and it also knows that these three words belong to the supersense "fruit" (this deterministic knowledge comes from an artificial ontology like WordNet). Thus, for the input "apple", the predicted embedding shall not only be closer to the actual "apple" sense embedding but also closer to the "fruit" embedding. After training, if we input an unseen word "durian", we will get nothing because we do not have the definition embedding for "durian". But we do have the universal supersense "fruit", and our trained model is supposed to put "durian" close to "fruit". Therefore, our model would predict that "durian" belongs to “fruit”. This can be regarded as low-level disambiguation. By using the WordNet supersense, which is much smaller than the vocabulary size, deterministic knowledge ("apple" belongs to "fruit") can be integrated to statistical embeddings, providing stronger reasoning and symbolic knowledge for our word sense model.

Test Results

Number of Iteration	Number of Training Data	Number of Dev Data	Accuracy for Known Words in Dev Data	Accuracy for Unknown Words (supersense level) in Dev Data
45	First 15000 of total 70000 examples	7118 (both known and unknown words)	62.9% (out of 6440 known words)	72.8% (out of 678 unknown words)

Future Directions

Constructing a baseline model: hard-code the model to always predict the most frequent sense for each word in the training set as the result. This will be a useful baseline performance when evaluating our model.

Providing more dataset statistics: for each ambiguous word, are the training examples evenly distributed over all its senses? If the training examples for a particular word is biased towards a particular sense, the model may not be able to catch the features of other senses. If this is true, it is reasonable to implement balanced class weight: the optimizer will punish the model more when seeing a sample with rare sense. This ensures the model to stop being biased towards more common senses and catch the features of some less frequent senses.

We should not be limited to only the WordNet supersense. We can integrate other complicated ontologies or hierarchy symbolic knowledge to statistical embeddings in the same way. This fusion may generate surprising results.

Using the correct dataset! As you can see in the dataset screenshot, I am using the lemmatized version of UDE. All “is”, “are”, and “am” are transformed to “be”. There are other kinds of information loss in the lemmatized version. After adapting to the correct original UDE version, the model is supposed to perform better.

References

Camacho-Collados, J. and Pilehvar, M. (2018). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. [online] arXiv.org. Available at: https://arxiv.org/abs/1805.04032 [Accessed 1 May 2019].

Neelakantan, A., Shankar, J., Passos, A. and McCallum, A. (2015). Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. [online] arXiv.org. Available at: https://arxiv.org/abs/1504.06654 [Accessed 1 May 2019].

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. [online] arXiv.org. Available at: https://arxiv.org/abs/1301.3781 [Accessed 1 May 2019].

Jeffrey Pennington, Richard Socher, Christopher D. Manning (2015). GloVe: Global Vectors for Word Representation. [online] https://nlp.stanford.edu/pubs/glove.pdf

Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli (2017). Neural Sequence Learning Models for Word Sense Disambiguation. [online] https://www.aclweb.org/anthology/D17-1120

Tang, G., Sennrich, R. and Nivre, J. (2018). An Analysis of Attention Mechanisms: The Case of Word Sense Disambiguation in Neural Machine Translation. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.07595 [Accessed 1 May 2019].

George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Universaldependencies.org. (2019). Universal Dependencies. [online] Available at: https://universaldependencies.org/ [Accessed 1 May 2019].

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018). Deep contextualized word representations. [online] arXiv.org. Available at: https://arxiv.org/abs/1802.05365 [Accessed 1 May 2019].

Mooney, J. Reisinger, J. (2010). Multi-Prototype Vector-Space Models of Word Meaning. [online] Cs.utexas.edu. Available at: http://www.cs.utexas.edu/users/ai-lab/?reisinger:naacl10 [Accessed 1 May 2019].

Kumar, S. and Tsvetkov, Y. (2019). Von Mises-Fisher Loss for Training Sequence to Sequence Models with Continuous Outputs. [online] Openreview.net. Available at: https://openreview.net/forum?id=rJlDnoA5Y7 [Accessed 1 May 2019].

Sepp Hochreiter, Jurgen Schmidhuber. LONG SHORT-TERM MEMORY. Neural Computation 9(8):1735-1780, 1997. [online] https://www.bioinf.jku.at/publications/older/2604.pdf

Ciaramita, M. and Johnson, M. (2003). Supersense tagging of unknown nouns in WordNet. EMNLP 2003. [online] https://dl.acm.org/citation.cfm?id=1119377

Chakaveh Saedi, António Branco, João António Rodrigues, João Ricardo Silva (2018). WordNet Embeddings. [online] https://www.aclweb.org/anthology/W18-3016

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P. and Robinson, T. (2013). One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. [online] arXiv.org. Available at: https://arxiv.org/abs/1312.3005 [Accessed 1 May 2019].