Abstract

In this paper, we investigate the problem of multiple word senses in different contexts. We build models to check if a given word is used in the same context in two different utterances, and evaluate these models on the recently released Word-in-Context dataset. Apart from baseline approaches using pre-trained GloVe embeddings, we also evaluate multisense probabilistic embeddings and explore recent neural representations. Our best model (fine-tuned BERT) has state-of-the-art performance on this dataset, and is currently ranked first on the WiC CodaLab competition leaderboard.

1 Introduction

A single word can have multiple interpretations in natural languages. For example, homonyms such as the word “spring” could be similar in one sense to “jump” and “vault”, but similar in another sense to “summer” and “winter”. A singular word representation captured by popular frequentist approaches such as GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2016) often fail to account for these multiple senses. Many generative and probabilistic models have been proposed for language as a whole, but only very recently have we begun to target generative models for embeddings. A probabilistic formulation incorporates uncertainty information and allows one to uncover multiple meanings with multimodal density representations. The investigation of such approaches is quite recent, arising from work described in (Athiwaratkun and Wilson, 2017) and (Athiwaratkun et al., 2018), and has been focused on the creation of these probabilistic models. Very recently, deep bidirectional pre-trained language representations such as those described in (Devlin et al., 2018) have given state-of-the-art performance on several natural language tasks such as question answering, paraphrasing and sentiment analysis, and we investigate these too in our context.

1.1 Problem Description

In our work we attempt to predict if a given word is used in the same sense in different utterances.

Figure 1: Simplified diagram of the prediction task

Figure 1 shows an instance of this prediction task, where we have determined that the target word “current” is used in a different sense in the two utterances.

2 Data

We use the Word-in-Context (WiC) dataset (Pilehvar and Camacho-Collados, 2018). The dataset contains 10000 sentence pairs with a common target word and a ground truth label indicating whether the intended sense of the target word is the same in both sentences or not. The dataset is balanced across the two classes.

This dataset is released as a part of an active CodaLab challenge, and as such the authors have not released the test set yet. The dev set currently serves as the test set and its labels are not public.

3 Evaluation Method

The competition uses accuracy as its measure of success, so we adopt the same as our primary metric. We split the competition train set into our own
train and dev sets (80/20 split), and use the competition dev set as our test set (we are not given the ground truth labels for this set). Henceforth, we will refer to the competition dev set as our test set (as opposed to our train and dev sets). We use the test set only for our best model (fine-tuned BERT).

The numbers shown in this report are based on our dev set performance.

To evaluate our classifiers holistically and independent of thresholding effects, we also measure the area under the ROC curve (ROC AUC) as another metric of performance. We believe that we may be able to achieve better accuracy using a well-chosen threshold.

Other than quantitative metrics we also look at qualitative strategies. Our predominant strategy is to identify which sentences and training examples the classifier makes a “large” mistake on. We find these by sorting the misclassification difference and reporting the worst performing results. We try to reason out why such a mistake could be made, tie it to the modelling strategy and then suggest possible improvements in modelling.

Ultimately, we generate predictions on the test set using our best model and upload them onto the competition portal, comparing our performance with the wider world.

4 Methodology

We tackle the problem in 3 ways:

- Build a baseline classifier which uses simple non-contextual methods and embeddings to evaluate whether there is a need to incorporate more results via Bayesian statistics.

- Set up more baseline classifiers, however using contextual embeddings with smart combinations to combine and understand them. Our goal is to see whether we can do better than the baselines we set forth, and hence paving the way for contextual embeddings.

- Compare it with the absolute current neural state of the art and see where are are (Devlin et al., 2018).

The code for our experiments is publicly available at https://github.com/sigtryggurk/aa228

4.1 Baseline Models

We evaluate three simple non-neural baselines. The first is a random classifier, which guesses randomly based on prior class probabilities (equal in our case). This is an absolute lower bound on performance and any reasonable model should be significantly better than this.

For our two actual baseline models, we use the following approach. We first create a 300-dimensional feature for the first utterance by averaging the GloVe embeddings (Pennington et al., 2014) for the words in the utterance. We repeat the same for the second utterance. Finally, we get another 300-dimensional embedding corresponding to the target word. We concatenate these three vectors to generate a 900-dimensional feature vector corresponding to each utterance pair and target word (i.e., each training example). We then use Logistic Regression (LogReg) on one hand, and Gradient Boosted Decision Trees (Ke et al., 2017) (using LightGBM) on the other hand to build classification models.

4.2 Experimentation

In this section we describe the setup for each experiment. Each experiment uses successively more complex representations and aims to mitigate shortcomings found in previous approaches. The results for each experiment are presented and discussed in section 5 below.

Experiment 1: Retraining GloVe on clean corpus

In an effort to improve performance and compare with other prior work, we retrain GloVe on the Westbury corpus (Shaoul and Westbury, 2010) and substitute it into our baseline models.

Experiment 2: Subword fastText

We substitute GloVe vectors with subword fastText representations (SubFT) described in (Athiwaratkun et al., 2018), that combine the n-gram subword embeddings of a given word into a single vector representation. We repeat the same procedure as in our baseline models (using Logistic Regression and Gradient Boosted Decision Trees).

Experiment 3: Multisense fastText

Rather than using a single subword fastText representation, we use vectors corresponding to both senses (MultiFT) as described in
We concatenate these vectors to produce a single 600 dimensional vector for each of our 3 components (sentence 1, sentence 2, target word), resulting in an 1800 dimensional vector that gets fed into our classifiers.

**Experiment 4: BERT**

Released a month or so ago, BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language representation that shows state-of-the-art performance on most NLP tasks (Devlin et al., 2018). BERT can be fine-tuned on specific tasks and has been found to be effective even with relatively small training corpora. BERT's model architecture is a multi-layer bidirectional transformer encoder which uses contextual information from the future and the past. This is an improvement from the OpenAI GPT model (Radford et al., 2018) which uses unidirectional information. BERT uses contextual information from both future and past words seen so far. The other main advantage of using BERT is that it has shown great performance for pairs of sentences including the SQuAD question answering dataset (Devlin et al., 2018) which uses unidirectional information. BERT uses contextual information from both future and past words seen so far. The other main advantage of using BERT is that it has shown great performance for pairs of sentences including the SQuAD question answering dataset and other similar sentence pairs datasets, and this was critical to our task. The input embeddings are the sum of the word embeddings, the segmentation embeddings and the position embeddings. We see here that it uses the traditional word embeddings.

We fine-tune a pre-trained BERT model for our specific task (using our training set) and evaluate its performance on our dev set. Specifically, we use an Adam optimizer with a learning rate of $2 \cdot 10^{-5}$, a batch size of 32, and train for 3 epochs on a k80 NVIDIA GPU.

After these experiments, we pick our best model (based on accuracy on our dev set) and use this to make predictions on our test set (i.e., the competition leaderboard dev set from CodaLabs). We submit this to the competition and evaluate our performance versus other teams.

### 5 Results

Table 1 below outlines the results of our experiments using the two performance metrics. Note that the last row shows the results of our best model evaluated on the test set (competition dev set), whose ground truth labels are hidden to us.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>ROC AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Classifier</td>
<td>0.493</td>
<td>0.493</td>
</tr>
<tr>
<td>LogReg (GloVe)</td>
<td>0.677</td>
<td>0.732</td>
</tr>
<tr>
<td>GBDT (GloVe)</td>
<td>0.703</td>
<td>0.764</td>
</tr>
<tr>
<td>LogReg (SubFT)</td>
<td>0.631</td>
<td>0.689</td>
</tr>
<tr>
<td>GBDT (SubFT)</td>
<td>0.719</td>
<td>0.776</td>
</tr>
<tr>
<td>LogReg (MultiFT)</td>
<td>0.664</td>
<td>0.709</td>
</tr>
<tr>
<td>GBDT (MultiFT)</td>
<td>0.706</td>
<td>0.768</td>
</tr>
<tr>
<td>BERT+fine-tune</td>
<td>0.865</td>
<td>0.917</td>
</tr>
<tr>
<td>(test)</td>
<td>0.707</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 1: Model performance. All numbers are reported on our dev set, except the final row that represents test performance (dev set on the CodaLab portal with unknown gold labels). We cannot calculate AUC scores for the last row as we do not know the actual gold labels.

The random baseline, as expected, gives a lower bound of about 50% on both accuracy and AUC. The GloVe-based baseline models perform much better, with GBDT showing an accuracy of 0.703 and AUC of 0.764.

Our re-trained GloVe embeddings perform slightly better than the pre-trained embeddings. However, we do not report these results here since the focus is on probabilistic embeddings and deep bidirectional language representations.

We then look at the results of probabilistic subword embeddings (SubFT). The LogReg models perform worse than with vanilla GloVe embeddings, but the GBDT-based models perform better. The multisense probabilistic embeddings (MultiFT) perform better than SubFT, validating our hypotheses that using multiple senses to model words helps improve performance.

However, these methods are left far behind by the performance of BERT fine-tuned to our corpus. We achieve much better results on both accuracy and AUC. We hypothesize that performance of this model on the test set may be state-of-the-art across all entries in the competition so far. This is in keeping with the performance of such models on other NLP tasks. It is also to be noted that we use non-neural models for our baseline as well as the SubFT and MultiFT embeddings, and hence BERT may appear to be much better than Mul-
tiFT than if MultiFT embeddings were used with a wide and deep bidirectional LSTM with attention.

Using our fine-tuned BERT model we generated predictions on the test set (CodaLab dev set) and uploaded them onto the competition portal so that we may compare our performance with the wider world. Figure 2 shows that our submission is currently in 1st place on the leaderboard, with 70.66% classification accuracy.

![Figure 2: Screenshot of the leaderboard of the CodaLab competition where we are currently in 1st place globally. See live scores at https://competitions.codalab.org/competitions/20010.](image)

It is interesting to note that the competition dev set results (i.e., our test set results) are much lower than those seen on our dev set. Furthermore, global results on the competition leaderboard are all much lower than the performance of similar models on our dev set (for context, we managed to get similar performance (on our dev set) with contextual embeddings and Gradient Boosted Decision Trees to deep and wide neural models on the global dev set, which seems surprising). To this end, we looked at the mean frequency rank of the target word in the competition’s train set (from where our train and dev sets are derived) and the competition’s dev set (our test set). We notice that the mean rank on the train set is 9546, while the mean rank on the competition dev set is 43842. Words in natural language generally follow a Zipfian distribution (inversely proportional to their frequency rank). This means that a lot of the target words in the competition dev set turn out to be rare words, which are often OOV (out of vocabulary) for most models. The fact that BERT was pre-trained on a large corpus might also be helping us on the competition.

5.1 Misclassified Samples

We conduct a hand-analysis of misclassified samples on our dev set to gain qualitative insights into the performance of our best model. Table 2 below shows the sample utterance pairs and how they were misclassified.

<table>
<thead>
<tr>
<th>Utterance 1</th>
<th>Utterance 2</th>
<th>Pred.</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you push it to the limit, <strong>safety</strong> is not guaranteed.</td>
<td>insure the <strong>safety</strong> of the children</td>
<td>diff.</td>
<td>same</td>
</tr>
<tr>
<td><strong>control</strong> the budget</td>
<td><strong>control</strong> an account.</td>
<td>same</td>
<td>diff.</td>
</tr>
<tr>
<td>She <strong>cut</strong> the deck for a long time</td>
<td>Wayne <strong>cut</strong>.</td>
<td>diff.</td>
<td>same</td>
</tr>
<tr>
<td>Where did you <strong>come</strong> from?</td>
<td>He <strong>came</strong> from France</td>
<td>same</td>
<td>diff.</td>
</tr>
</tbody>
</table>

![Table 2: Hand-curated samples of misclassified utterance pairs showing the predicted and the true labels. The target word is shown in bold in each utterance pair.](image)

In the first example in the table, we see that the contexts for the two utterances are very different, and since our approach relies on the context words, it’s unsurprising that our approach fails to correctly classify the sense of the target word when the context serves as a red herring. Similarly, in the second example the contexts are very similar, but the senses are different.

In the third example we have very minimal context in the second utterance, that is completely distinct from that of the first utterance.

Some of the issues arising from misleading context could be alleviated by considering the syntactic features as well. For example, if the model knew that the word ‘cut’ in the third example was a verb in both cases, it would stand a better chance of recognizing that the word is used in the same sense in both utterances.

In the fourth example, the model likely struggles when the stem of the word changes, and does not capture that these are in fact the same word. This kind of misclassification was rare, but could
easily be mitigated by first using an appropriate stemming algorithm (e.g. the Porter Stemming algorithm (van Rijsbergen et al., 1980)) before training. As an aside, we question the validity of the ground truth label in this particular example. In our opinion the verb ‘come’ is used in the same sense in both utterances.

In the last example we likely suffer from a data sparsity problem in the word embeddings. Using the word ‘retrograde’ in the sense of recapping is rare and completely eclipsed by its usage in astronomy/astrology.

6 Conclusion

We see that there are two levels of abstraction in this problem. The first is the word (or sub-word) embeddings which convert a word space into a vector space of numbers. The second is the sentence-level embedding which involves understanding the combinations of different words and coming up with a representation for sentences.

Based on our experiments, we make the following remarks:

- Contextual embeddings help improve performance over general-purpose embeddings, and can be used in situations where context matters.

- Context is hard to capture with most current approaches (using non-neural models as well as neural ones), given the inherent ambiguity seen in language.

- Large state-of-the-art pre-trained models can be adapted to specific contextual tasks even with a small amount of training data. As an aside, the inclusion of context primarily affects the last few layers of the network, and hence we can think of smarter methods to incorporate contextual information into these models.

7 Future Work

A comprehensive study on word embeddings versus contextual ones is no easy task. Not only do we need to consider many embedding assumptions, we also need to look into the different models these embeddings are being used with. We can think of the following avenues for future work:

- As we mentioned earlier, our current work uses non-neural models to evaluate embeddings and is hence not the best possible comparison with state-of-the-art (such as BERT). A good avenue for future work would be to evaluate the performance of these embeddings on neural models, e.g., bidirectional LSTMs with attention.

- We could train a language model from scratch based on contextual appearance of words, and use it to predict if both sentences in the sentence pair use the target word in the same context.

- We could explore the idea of unsupervised sentence representations (versus averaged word vectors) such as sent2vec (Pagliardini et al., 2017) and Universal Sentence Encoder (Cer et al., 2018), and evaluate the possibility of incorporating context into these.

- Currently, there is no framework to deal with generative models for word embeddings. One possible (and potentially promising) route would be to mathematically formulate a framework for generative word embeddings and evaluate these on similar context-sensitive tasks.

References


Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strupe, and Ray Kurzweil. 2018. Universal sentence encoder.


Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan


