University *of Ljubljana*
Faculty *of Computer and Information Science*

Cross-lingual sense disambiguation

Zala Erič, Miha Debenjak, and Denis Derenda Cizel

Abstract

Depending on the context the same word can refer to several possibly unrelated meanings. Simple word embedding methods (Word2Vec, GloVe ...) cannot always reflect the dynamic semantic nature. The BERT model attempts to overcome this shortcoming by computing dynamic word representations that adapt according to context. Our task was to produce a corpus of sentence pairs with the same words that can be used in different senses and then to classify same words in different sentences into similar or different senses using the BERT model for Slovene language.

Keywords

NLP, sense disambiguation, word-in-context, BERT, SloBERTa ...

Advisors: Slavko Žitnik

Introduction

In human language, the same word can have different meanings depending on its intended use in a sentence. In the context of natural language processing, word sense discrimination can be described as the ability to determine the meaning of a word used in a given sentence or context. We know word syntactic ambiguity, which can be solved by part-of-speech (POS) taggers, and semantic ambiguity (word sense disambiguation), which is part of our project, and it's harder to resolve than syntactic ambiguity. Word Sense Disambiguation (WSD) has to decide what the sense of a word is based on the word in context and a fixed inventory of potential word senses. This principle can be used in various applications: informational retrieval, knowledge acquisition and information extraction, machine translation, question answering, lexicography. Our goal is to prepare a solution that will be able to determine if the word X is used in the same context or not in 2 different sentences.

We decided to classify the meaning of words using BERT models for Slovene language. Since a database with sentence pairs for our language is not available/does not exist, we first searched for suitable words, their uses in sentences and then manually classified part of the collected data. We then ran 2 different classification approaches using different Bert models on the manually annotated data to be able to automatically classify all the other sentence pairs in the corpus. The whole workflow is described in detail later in the report.

Related work

To better understand the task itself, we reviewed similar work in the research area. The scope of the thesis is in general well researched. Through a detailed analysis of similar works, we have found the most appropriate procedures that could be applied to the Slovene language. Below are summaries of papers and their methods, which are relevant to our work.

WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representation [1]

Word-in-Context is a dataset made for the binary classification task of determining whether a word is used in the same context in two different sentences. It was built from usage examples from three lexical resources: WordNet, VerbNet and Wiktionary. Each instance of the database consists of the target word and two sentences in which the word is used. The instances can be positive, meaning the word is used in the same context or negative, meaning the words are used in different contexts. It is already divided into the development and test subsets of sizes 1600 and 800 instances respectively. The authors made sure that the splits were balanced in terms of positive and negative instances and that there were at most 3 instances of the same target word, as well as not duplicating the context sentences. As some words have a lot of different meanings, the database does not include senses which are very similar to each other. The database was used with a series of most performing WSD methods, with the BERT method being the best performing on the new dataset with the accuracy of 65,5 %.

SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems [2]

The GLUE benchmark is a single-number metric that summarizes progress on a set of language understanding tasks. Since performance of recent methods (ELMo, OpenAI GPT, BERT) on the GLUE benchmark has recently surpassed the level of non-expert humans, the authors present SuperGLUE, with which they aim to pose a more rigorous test of language understanding. SuperGLUE consists of a public leaderboard built around eight language understanding tasks, drawing on existing data, accompanied by a single-number performance metric and an analysis toolkit. The eight tasks are the following: Boolean Questions, Commitment Bank, Choice of Plausible Alternatives, Multi-Sentence Reading Comprehension, Reading Comprehension with Commonsense Reasoning Dataset, Recognizing Textual Entailment, Words-in-context, Winograd Schema Challenge, and they are evaluated equally to produce a single-number score. They are compared to a human performance. The authors evaluated BERT-based baselines and found that they still perform worse than humans by nearly 20 points. Given the difficulty of SuperGLUE for BERT, they expect that further progress in multi-task, transfer, and unsupervised/self-supervised learning techniques will be necessary to approach human-level performance on the benchmark.

Knowledge-based Word Sense Disambiguation using Topic Models [3]

Most of the word sense disambiguation systems use the sentence in which the word is used to determine its sense. But we know that the whole text which includes this sentence is very likely to cover a specific topic of some wider domain and the sense of words used in this domain are likely to be similar. That is why Chaplot and Salakhutdinov introduced a new word sense disambiguation method, which incorporates topic modeling to detect the topics used inside the document and use these topics to connect words to their senses. The topic modeling method used in their research is the Latent Dirichlet Allocation (LDA), which is an unsupervised method which we can initialize with specific priors. In this case the LDA method is initialized with priors from the synset set of synonyms from the WordNet lexical database. The model then returns the sense of the word which is the most probable in each document in the corpus. It assumes that all occurrences of a word in a document are used in the same sense, which is an unrealistic assumption.

Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations [4]

This article presents different strategies of integrating pre-trained contextualized word representations for WSD. Contextualized word representations are effective in downstream natural language processing (NLP) tasks, such as question answering, etc., but on word sense disambiguation (WSD), it does not outperform the state-of-the-art approach that uses noncontextualized word embeddings. The contextualized word representation used is BERT (Devlin et al., 2019), a bidirectional transformer encoder model (Vaswani et al., 2017) pre-trained on billions of words of texts. The model is trained on two tasks, masked word and next sentence prediction, where in both, prediction accuracy is determined by the ability of the model to understand the context. The authors describe different techniques of leveraging BERT for WSD, broadly categorized into nearest neighbor matching and linear projection of hidden layers. The tasks for evaluating are the English all-words task, English lexical sample task, and Chinese OntoNotes WSD task. The best results were consistently obtained by linear projection models. The final results of the authors' best BERT WSD models greatly outperformed previous neural WSD models.

Methods

Roughly speaking, the work of our term paper can be divided into 2 parts: the preparation of the data and the creation of the corpus, and then the preparation and use of a simple classifier.

Data preparation

As there is no Slovene database similar to the WiC database for evaluation of word sense disambiguation methods, a new database was constructed. The steps to create this database are similar to those proposed in the WiC article. First, we manually selected and transcribed 250 different words from the "Dictionary of Slovene Homonyms", published in 1997, for which we were able to determine different meanings of usage. The problem was that many of the words in the collection are antique or dialectal, so we did not use them. Then the usage examples for target words were obtained from slWac 2.1 corpus, a web corpus collected from the .si top-level domain. Published version of it on SketchEngine contains 895 000 000 tokens. We then created a database where we collected pairs of 2 different sentences for each word. 250 of these sentence pairs were manually checked to determine whether the target word is used in the same or different senses in each case. We made sure that the ratio of similar and different meanings in the manually annotated part of the database was even.

Analysis

According to the articles described in the Related Work section, the BERT method woule be suitable for our analysis part of project. BERT, which stands for Bidirectional Encoder Representations from Transformers, is an ML framework for NLP. BERT was pretrained on two tasks: language modelling (15 % of tokens were masked and then it was trained to predict them from context) and next sentence prediction (BERT was trained to predict if a chosen next sentence was probable or not given the first sentence). As a result of the training process, BERT learns contextual embeddings for words. After pretraining BERT can be fine-tuned on smaller datasets to optimize its performance on specific language processing tasks. BERT was developed by Google and is also used in

Google search engine to optimize the interpretation of user search queries.

Classification with SloBERTa

In one of our approaches, based on [5], we tokenized two sentences using the pre-trained `SloBERTa` AutoTokenizer. They both contain the homonym that we want to disambiguate. We combine both sentences and add a special token `'[CLS]'` to the beginning of the first sentence and `'[SEP]'` to the end of the first sentence (before the first token of the second sentence) to separate the two. We then converted the tokenized text from a list of strings to a list of vocabulary indices. We observe here that our token has the same index. We then make a mask for the tokenized text where every token of the first sentence including `'[CLS]'` and `'[SEP]'` is represented by a 0, and every token of the second sentence is represented by a 1. This is because BERT expects sentence pairs, using 1s and 0s to distinguish between the two sentences.

We then convert our data to torch tensors and call the pre-trained `SloBERTa` model. We set it to evaluation mode and retrieve the hidden states of the network. It has the following four dimensions:

- number of layers: 13, initial embeddings + 12 BERT layers,
- number of batches: 1,
- number of tokens: num_tok = len(tokenized_sent1) + len(tokenized_sent2) + 2, for the two special tokens,
- number of hidden units: 768.

We only want dimensions tokens, layers and features so we use the `permute` function from `PyTorch`. Next, we want to somehow get a single vector to represent a token instead of all the 13 vectors that we have at this step for each token. We can use concatenation of the last six layers, giving us a single word vector per token. Each vector will have length 6*768 = 4608, giving the total shape of token vectors num_tok x 4608. We can also opt for summing the last six layers together, giving the total shape of token vectors num_tok x 768.

To confirm whether the value of these vectors are contextually dependent, we can now extract the cosine similarity of the vectors at indexes of our tokens by using `cosine` of the `SciPy` library. An example of the structure of such a classification procedure is shown in Figure 1.

We also tried comparing 2 sentences with the same word directly against each other. We loaded the model and tokenizer using AutoTokenizer and AutoModel and tested different BERT models. We then tokenized all the words and calculated the similarities between all the words in the given sentences using cosine similarity. This was done in order to extract more information about the context itself from all the words in the sentence, as it has proved very difficult to predict the similarity of the contacts of sentences based on the comparison of a single word in certain cases.

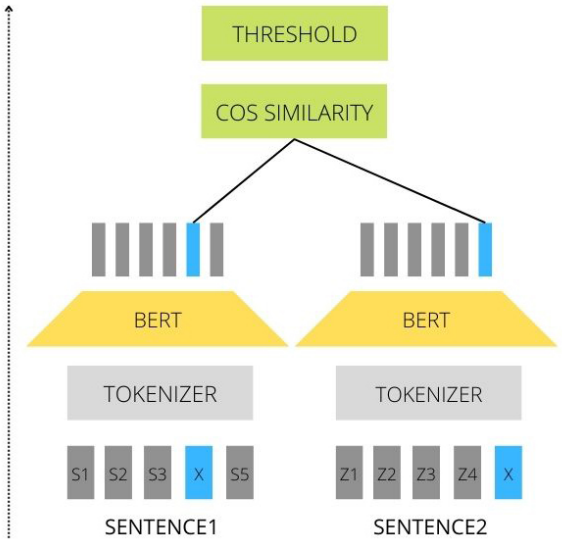

Figure 1. Structure of the application of the BERT model

Classification by masking

In the second approach we used the fill mask BERT pipeline to retrieve words in the spot of our target word and then compared the retrieved words of two sentences. The model fills the mask with words, which are more likely to appear in the given spot based on the context it gets from the remaining words in the sentence. That is why this feature can be used in word sense disambiguation task. If the retrieved words from the first sentence are similar to those from the second sentence, we can say that the target word is used in the same meaning in both sentences. If the words are strongly different, we can say that the meaning is not the same in the two sentences. We used two different BERT models for the mask fill task: SloBERTa [6] and CroSloEngual BERT [7].

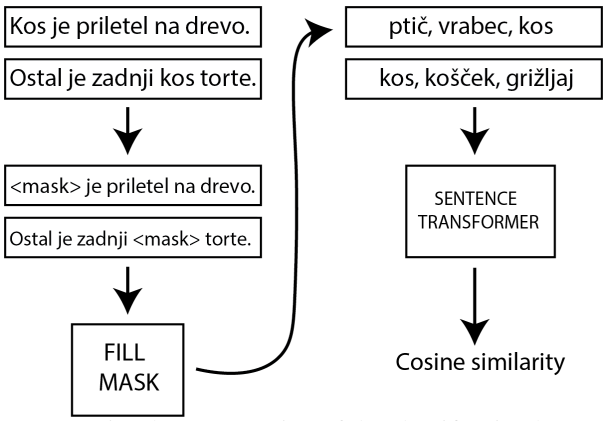

Figure 2. Visual representation of the classification by masking pipeline.

To compare the retrieved words of the fill mask task we used BERT sentence transformers. We pretend that the retrieved words are actually sentences and embed them using sentence transformers. Similar to the base BERT encoders, the BERT sentence transformer encodes whole sentences into

embeddings. We compare the obtained sentences by calculating the cosine similarity of their embeddings. For this task we used the use-cmlm-multilingual model form the sentence-transformers library.

Results

The results section is also divided into two parts. The results of each methodology are described below.

Dataset

We've compiled a list of 250 different slovene words that can be used in multiple senses. We then used these words to produce 2000 pairs of sentences, resulting in 4000 sentences of use of these words in the texts. We collected sentences using provided API interface for SketchEngine. Due to the limitations of the free API account, we had to retrieve the sentences in several queries over a longer period of time.

The data is stored in a format as: *token, sentence1, sentence2, tokenid1, tokenid2* in the *corpus.csv* file. The token represents the observed word, sentence1 and 2 an example of the use of such an observed word in a sentence, and tokenid1 and 2 the position of the observed word in each of the sentences.

A separate file *corpus-annotated.csv* is also available, which has the following format: *token, similarity, sentence1, sentence2, tokenid1, tokenid2*. The value of similarity represents whether the token has the same or a different meaning in the two examples of sentences, depending on the context. A value of 1 for similarity represents the same meaning and a value of 0 represents a different meaning of the word. All other columns have the same meaning as the basic corpus described above.

The file *auto-annotated.csv* represents the results of the automatic classification of sentences. The last column holds information if the two uses of the word are for the same or different contexts.

Evaluation of classification mathods

We evaluated our methods with different metrics, accuracy and precision. Accuracy is the percentage of all correctly predicted cases, computed as:

$$accuracy = TP + TN/TP + FP + TN + FN$$

.

Precision tells us how precise the model when predicting a sample as positive, computed as:

$$precision = TP/TP + FP$$

.

For some methods, we include comparisons of different models in this task.

Classification with SloBERTa

The results of our classification with SloBERTa turned out okay. Firstly and most importantly, SloBERTa divided quite a number of our homonyms into subwords. It didn't include some method to combine the vectors of the subwords into one to compare their distances. In one experiment, we simply neglected the divided words when it came to determining the threshold. With this tactic, our threshold was 0.77 for the summation method (SUM) and 0.79 for the concatenation method (CAT) (both described in the chapter). With such thresholds, we reached the accuracy of 56.3% and the precision of 54.2% in SUM and the accuracy of 60.5% and the precision of 65% in CAT.

This called for some upgrades, so we tried to also include a tactic that mitigated the effect of being unable to process the token as whole. We added some simple code to detect when our token was split, and when we find this case, we compute the cosine similarity for both parts of this word and then average it. This implementation only handles our word being split in two parts, as for more the code would have to be changed quite a lot to be more dynamic. Our threshold was really hard to determine here, as the cosine similarities between the two classes were closer now. The threshold was 0.77 for CAT and 0.78 for SUM. With such thresholds, we reached the accuracy of 54.3% and the precision of 51.6% in CAT and the accuracy of 59.6% and the precision of 69.4% in SUM.

We also tried different models, such as CroSloEngual and BERT-base-multilingual-uncased, but the results were similar as it made the division of words slightly worse in some cases.

Moving on to the next example of using the BERT model, table 1 below shows the results for comparing complete sentences against each other. We have used 3 different BERT models and as is obvious, their performance is quite different. The difference from the baseline implementation is that we now compare similarities for all words, not just the observed word.

BERT model	Threshold	Recall	Accuracy
CroSloEngual	0.53	61.5 %	64.0 %
SloBERTa	0.86	53.2 %	62.2 %
BERT-base-multil.	0.74	57.9 %	54.7 %

Table 1. Performance of different models

Classification by masking

We used the classification by masking on the manually annotated corpus in order to measure its accuracy. By repeating this with different numbers of words the mask fill returns we obtained a number of newly artificially annotated corpora with the cosine similarity attached. To get a feeling for the threshold of the classifier, we calculated means of instances which were marked of the same meaning and those marked as different meaning. We then classified the instances by thresholds ranging from the mean of distances of same meanings to the mean of distances of different meaning.

Threshold	Accuracy
0.5	66,2 %
0.55	66,7 %
0.6	70,1 %
0.65	70,5 %
0.7	71,3 %
0.75	65,9 %
0.8	62,7 %

Table 2. Effect of the threshold on the accuracy of the classifier.

As we can see in Table 2 the accuracy tends to be better, when the threshold is closer to the mean of the distances of the instances with different meaning. We observed during the calculation of the distances, that distances of the different meaning instances are constantly higher, while the distances of the same meaning instances are sometimes correctly low, however in some cases they are also in the different meaning range. That is why the threshold has to be closer to the mean of the different meaning instances.

Next we tested how the number of words the fill mask task returns impacts the classification. The more words we provide to the sentence transformer, more context it can extract. However too much words might produce noise, as the words are becoming less and less relevant, since the fill mask task returns weighted words. The results in Table 3 imply that sentences consisting of 15-20 words give the best results. With more than 20 words the accuracy starts to fall, most probably due to the fact, that words are getting less relevant.

BERT model	No of words	Threshold	Accuracy
SloBERTa	5	0.7	64,3 %
	10	0.7	67,8 %
	15	0.65	69,8 %
	20	0.7	71,3 %
CroSloEngual	10	0.7	67,0 %
	15	0.7	68,6 %
	20	0.6	66,7 %

Table 3. Performances of models using different numbers of retrieved words for sentence comparison.

All the discussed results were obtained with the SloBERTa model. We also implemented the pipeline with the CroSloEngual BERT model and tested it in a similar way. As we can see in Table 3 the CroSloEngual model, which is a multilingual model trained on Croatian, Slovene and English corpora, achieved slightly worse results, but approximately in the same range. This is likely due to the fact that SloBERTa is a monolingual model made for Slovene language only and it has better embeddings for the Slovene language.

The final model uses the SloBERTa model and retrieves 20 words, while the threshold is 0.7. The sentence transformer model used is use-cmlm-multilingual [8]. It scored an accuracy of 0.713 on the manually annotated word in context corpus. After running the classification on the whole corpus with 2000 instances, it classified roughly half instances as same meaning and the other half as different meaning.

Discussion

Both parts of the assignment produced good results. It turned out that creating a corpus is a more challenging task than we had initially imagined. It is already quite difficult to select a large set of words in the Slovene language, to which we can attribute different meanings in different contexts. The only publicly available source of such words is a physical collection, published in year 1997.

For the classification itself, we have chosen 3 different methods, based on the same basis but producing different results due to the different ways of applying it. The performance is to be expected, as we have already had a lot of problems comparing word meanings when manually annotating sentences.

The final model performs poorly on very short and very long sentences. It is difficult to detect context even for humans in short sentences, as they tend to lack meaningful words. In longer sentences there can be multiple context, which also do not bring useful information to our model, but rather make its work more difficult.

It is very likely that further upgrades and fine-tuning of the BERT models could deliver even better results. However, a large part of the difficulty was caused by the fact that we were not able to load the SloBERT model using the BertModel library, which is mostly used in the implementations of similar tasks.

References

[1] Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representation. 2019.

[2] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537, 2019.

[3] Devendra Singh Chaplot and Ruslan Salakhutdinov. Knowledge-based Word Sense Disambiguation using Topic Models. 2018.

[4] Hadiwinoto Christian, Ng Hwee Tou, and Wee Chung Gan. Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, page 5297–5306, 2019.

[5] Chris McCormick. Bert word embeddings tutorial, May 2019. [Online; posted 14-May-2019].

[6] Matej Ulčar and Marko Robnik-Šikonja. Slovenian RoBERTa contextual embeddings model: SloBERTa 2.0, 2021. Slovenian language resource repository CLARIN.SI.

[7] M. Ulčar and M. Robnik-Šikonja. FinEst BERT and CroSloEngual BERT: less is more in multilingual models. In P Sojka, I Kopeček, K Pala, and A Horák, editors, *Text, Speech, and Dialogue TSD 2020*, volume 12284 of *Lecture Notes in Computer Science*. Springer, 2020.

[8] Ziyi Yang, Yinfei Yang, Daniel Cer, Jax Law, and Eric Darve. Universal sentence representation learning with conditional masked language model. pages 6216–6228, November 2021.