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

Cross-lingual sense disambiguation

Jure Tič, Nejc Velikonja, and Sandra Vizlar

Abstract

This article describes our approach to Word in Context natural language processing problem. The existing work on the topic was researched and summarized. We followed those examples and used three models for classification: Word2Vec, SloBERTa and fastText. We followed similar corpus preparations methods as those described in related works, which were both manual and automated. We then analysed the models' outputs and compared them amongst each other to determine which was the best at determining the context.

Keywords

detect context, manual corpus annotation, automatic corpus annotation, Word2Vec, fastText, SloBERTa

Advisors: Slavko Žitnik

Introduction

In this paper we will attempt to tackle one of the most difficult areas of natural language processing, this being context recognition. Unlike in machine language, in natural language the same word can bear different meanings, depending on the context in which it was written or spoken. Such a word is called a homonym and can be very tricky for machines to recognize, as it often requires an advanced understanding of the language or even specific topic in which it was mentioned. Nevertheless recognizing context is very important because it enables us to make more informed decisions.

Our task is to prepare a Slovene corpus and then use a model to determine whether or not a chosen word is used in the same context in two different sentences. We will try to implement the Slovene version of the WiC SuperGLUE[1] task that will classify the contexts of the word in sentence pairs bearing the same meaning as true or false.

Related work

We have read multiple studies that have been done on the topic of word in context classification, each model achieving different results.

WiC dataset

The most relevant study on the subject was *WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations*[2].This article describes the process of preparing and optimizing a corpus for the English variant of the WiC task.

The article first described the structure of the WiC dataset as instances that have a target word and two sentences containing the word in a certain contexts. The task was to determine whether or not the two contexts are the same. They used a binary classification. When building the dataset they paid attention to the balance between positive and negative examples. They then performed a quality check on 400 instances manually.

The article then suggests different methods, such as Context2Vec, BERT and many others with which to solve the task of context recognition. Of the suggested methods BERT proved to be the most successful.

TransWiC

The second study *TransWiC at SemEval-2021 Task 2: Transformer-based Multilingual and Cross-lingual Word-in-Context Disambiguation*[3] improved upon its predecessor. They employed transformer models, that learn context and by that meaning by tracking relationships in data such as words in a sentence.[4] With those they also planned to minimize the dependency on a specific language by not using language-specific processing. They succeeded in achieving around 90% classification accuracy for the English language.

SkoltechNLP

In the article *SkoltechNLP at SemEval-2021 Task 2: Generating Cross-Lingual Training Data for the Word-in-Context Task*[5], authors chose to use a neural system based on the XLM-R (a pretrained transformer model).

XLM-R is pre-trained on a text in 100 languages, which enables cross-lingual classification. That means it can perform

cross-lingual understanding without additional training data, that wasn't even provided for the SemEval-2021 Task2, as the authors state. They conducted their experiments on multiple languages reaching the classification accuracy of 60% - 89%, depending on the language.

SuperGLUE

While not discussing the solutions to the context detection problem, SuperGLUE [1] is a framework for evaluation of Neural Language Processing tasks with the aim of encouraging improvement in their solutions. It is made for English language and features 8 tasks in which improvements are yet to be made. One of these tasks is also Word in Context.

Proposed Methods

We will use the article *WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations*[2] as baseline for our work. We will prepare the dataset in a similar way, also making it a binary classification problem.

We will use Gigafida[6] database to extract our sentences from. Gigafida is a collection of texts in the Slovenian language. It contains materials such as newspapers, literary texts, online texts and more. The corpus is reliably annotated and documented.

The first proposed method for automatic classification is Word2Vec. An algorithm that, using a two layered neural network, represents words as vectors and uses these representation to calculate similarity between words. [7]

The second method we used is FastText. A method that extends the Word to Vector approach with sub word information, making this method better at understanding prefixes and suffixes. [8][9]

The last method we used was BERT, more specifically SloBERTa [10]. This is a RoBERTa [11] model trained on slovene data, using transformer architecture to learn similarities and context between words.

In the article[2] they used BERT, which returned best results and since we will be working with sentences in Slovene, we have decided to use SloBERTa 2.0[10], a Slovene version of the model. We will compare FastText, Word2Vec and SloBERTa approaches.

Corpus preparation

To prepare the corpus we used the words from Slovar slovenskih homonimov[12], a dictionary of Slovenian homonyms. We combed trough it and picked ones that weren't too archaic and had most distinct meanings. The words we chose are seen in figure 2.

For every word we searched Gigafida database for sentences that contain it or a form of it. We obtained sentences containing the desired words with web scraping the database using a Python package called BeautifulSoup[13]. The script paired sentences containing the given word together randomly. We saved these pairs in a .cvs file. Each sample in the corpus is therefor represented by:

- the class of the sentence pair (that tells us if the context of the word in the sentences is the same),
- the word we are checking the context of and
- the sentence pair containing the word.

We annotated the acquired sentence pairs manually. Each sentence pair is labeled with a class that tells us whether the context of the word in both sentences is the same or not. We annotated the sentences with the context of the word being the same in both sentences as belonging to class 1. If the context was different we annotated the pair as belonging to class 0. We annotated 564 sentence pairs.

To test the quality of the classification models we also built a corpus for automated annotation, that the models will perform. We generated 5000 sentence pairs, which we also acquired from Gigafida using the same word list as before.

This automatic corpus was annotated using k-means clustering based on TFIDF scores. It is however not manually checked for validity of its annotations.

Its statistics can be seen in figure 1

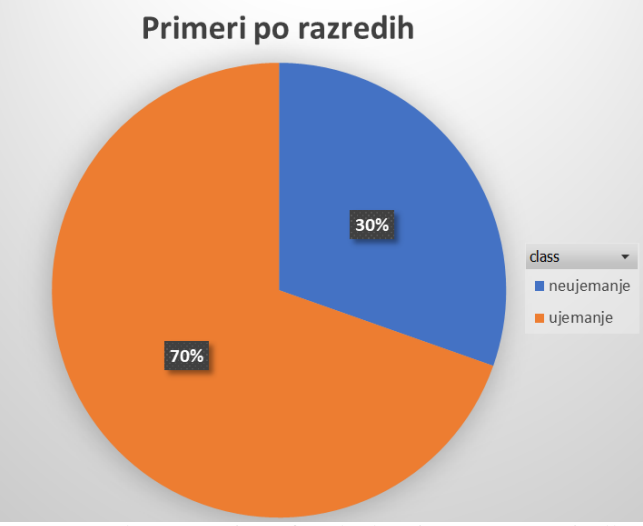

Figure 1. The proportion of each class in our automatically annotated corpus

Corpus analysis

Our manually annotated corpus, described in the above subsection "Corpus preparation", consists of 564 manually annotated pairs of sentences. The words we used can be seen on figure 2. Figure 2 also shows how many pairs of sentences we classified as 1 (having the same context) and 0 (the context being different) for all the words we used. As we can see in figure 3 on average there are more sentence pairs classified as having the same context at 58% and 42% being classified as having different context.

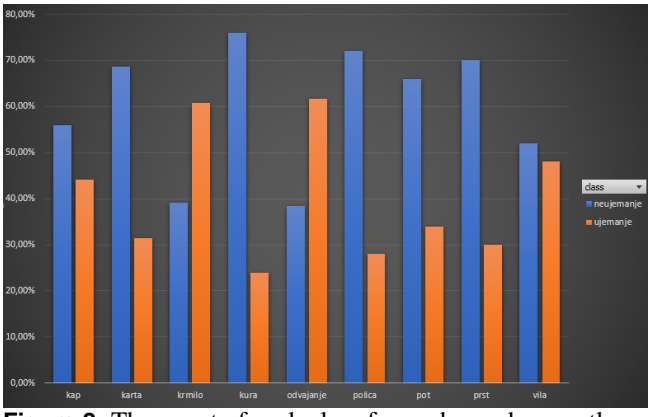

Figure 2. The count of each class for each word currently present in the corpus

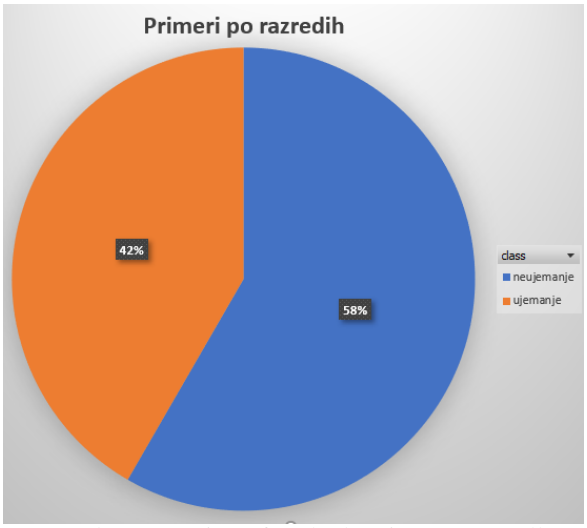

Figure 3. The proportion of each class in our manually annotated corpus

Experiments

Our approach to automatic sentence annotation was based on 3 different approaches, two were non-contextual embeddings and the last one was contextual.

Word2Vec

We first used a word2vec [7] approach. We used a pretrained slovenian word2vec model provided by NLPL (Nordic Language Processing Laboratory) [14] in context of this paper [15]. We then preprocessed the text by converting it to lower case and removing punctuation. The embedding of a sentence was computed by averaging all the embbeding vectors of all words in the example. If the word was not present in the embeddings we ignored it in this calculation. The final vectors for each of the two sentences were then compared by calculating the cosine similarity. We then classified the sentences based on the distance between them.

For this we had to determine the threshold, which we did by running the same algorithm on a manually annotated corpus, which is explained in detail in section "Corpus preparation". We collected the results for positive and negative sentences and calculated the mean and standard deviation of the results. We then put the threshold between the mean values of the two classes.

FastText

The second approach we used is FastText [8][9]. We used a pretrained model with embeddings in slovene [16]. Preprocessing was not required in this case, since the model supported constructing vectors from subword information. Similarly to Word2Vec approach we constructed the sentence vectors from the whole sentence. We used the method provided by FastText library for this calculation. This calculation is the mean value of vectors divided by their L2 norm (L2 norm has to be positive for a vector to be included in the calculation). We then compared the final sentence vectors by cosine similarity. The examples were annotated based on the threshold we determined by using the same approach on the manually annotated corpus.

SloBERTa

For the last approach we used a RoBERTa model [11]. We used a pretrained model on slovene corpus SloBERTa [10]. We converted all text to lower case since it eased the search for the word in the sentence considerably.

We first tokenized the sentences. The RoBERTa model tokenization is based on the byte pair encoding [17]. As such we had to determine where homonym is in the encoded vector. This can be hard since not all words are represented as whole and can be split into parts if they are not present in the model's vocabulary. We made this easier by using lemmatization only for the word search (the encoded sentences weren't lemmatized). Even with this approach we still had a few examples where the model could not find the word, in these cases we could not calculate an embedding and as such the distance was 0 and thus classified as same meaning. The same as in previous two approaches we determined the threshold by running the algorithm on the manually annotated data.

Results

As mentioned previously we used three different methods for detection of context. These are Word2Vec, FastText and SloBerta.

We evaluated their performance with Precision, Recall and F-score.

Precision is a metric that shows what percentage of positive identifications are actually correct. In our example, what percentage of identifications that the two words are used in the same context the words were in reality used in the same context. It is calculated by the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall on the other hand tells us what percentage of the cases that are really positive were identified correctly by the algorithm. It is calculated by the following formula:

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

The last metric we used was F-score. Another common way to measure accuracy, F-score us accuracy calculated from recall using the formula bellow:

$$F_1 = 2 * \frac{Precision \cdot Recall}{Precision + Recall} \tag{3}$$

After annotating our corpus with the methods described above, we got results as shown in the Figure 4

We also presented the results of our predictions in confusion matrices, that can be seen on figures 5,6 and 7

	Precision	Recall	F-score
Word2Vec	0.5998	0.6024	0.5943
Fasttext	0.5633	0.5596	0.5319
Bert	0.5811	0.5821	0.5690

Figure 4. The results of our experiment using the three methods

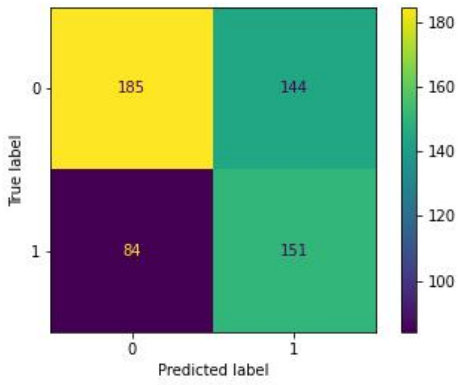

Figure 5. The confusion matrix of the examples annotated with the Word2Vec algorithm, 0 meaning a prediction of different meanings of the word in question, and 1 of the same

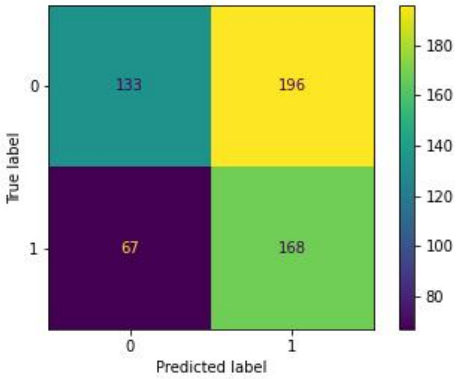

Figure 6. The confusion matrix of the examples annotated with the FastText algorithm, 0 meaning a prediction of different meanings of the word in question, and 1 of the samecc

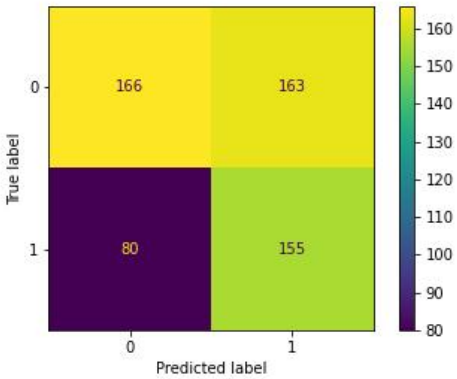

Figure 7. The confusion matrix of the examples annotated with the Bert algorithm, 0 meaning a prediction of different meanings of the word in question, and 1 of the same

Discussion

The task was proven to be very difficult for all models with all results being fairly similar, scoring roughly around 0.6 in all metrics. The best results were achieved by Word2Vec, but this could be contributed to chance, since the difference is small.

When comparing our results to that of other researchers we find many similarities, with another article researching the same field [2] reaching a 65 percent accuracy with BERT model and 60 percent accuracy with Context2Vec model.

This only underlines the fact that the accuracy of prediction in this field is very low and further research into the topic of context detection is certainly necessary.

References

[1] 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, 2019.

[2] Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Hansi Hettiarachchi and Tharindu Ranasinghe. Transwic at semeval-2021 task 2: Transformer-based multilingual and cross-lingual word-in-context disambiguation. *CoRR*, abs/2104.04632, 2021.

[4] What is a transformer model? — nvidia blogs. https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/. (Accessed on 05/23/2022).

[5] Anton Razzhigaev, Nikolay Arefyev, and Alexander Panchenko. SkoltechNLP at SemEval-2021 task 2: Generating cross-lingual training data for the word-in-context task. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 157–162, Online, August 2021. Association for Computational Linguistics.

[6] Tomaž Erjavec Miha Grčar Peter Holozan Simon Šuster Nataša Logar Berginc, Simon Krek. Gigafida, 2021.

[7] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[9] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

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

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[12] J. Bálint. *Slovar slovenskih homonimov: na podlagi gesel Slovarja slovenskega knjižnega jezika*. Razprave Filozofske fakultete. Znanstveni inštitut Filozofske fakultete, 1997.

[13] Bo Zhao. Web scraping. *Encyclopedia of big data*, pages 1–3, 2017.

[14] Word2vec embeddings for slovene. http://vectors.nlpl.eu/repository/. (Accessed on 05/24/2022).

[15] Andrei Kutuzov, Murhaf Fares, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 58th Conference on Simulation and Modelling*, pages 271–276. Linköping University Electronic Press, 2017.

[16] Nikola Ljubešić and Tomaž Erjavec. Word embeddings CLARIN.SI-embed.sl 1.0, 2018. Slovenian language resource repository CLARIN.SI.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.