FRI Natural language processing course 2022
Project report

University of Ljubljana
Faculty of Computer and Information Science

Cross-lingual sense disambiguation

Erazem Pušnik, Rok Miklavčič, and Aljaž Šmaljcelj

Abstract
TODO

Keywords
Natural language proccessing, word context, word sense disambiguation

Advisors: Slavko Žitnik

Introduction

In human language, one word can often be used in multiple ways. It is important to understand word usage to properly develop natural language processing applications. While the problem is easy for humans to solve, since we know from experience of how the world works, computers aren't able to naturally do so.

Word Sense Disambiguation is an important method of dealing with a word context and it involves the use of syntax, semantics and word meanings in context. It can be used in multiple applications, like machine translation, speech processing, information retrieval, text processing and more. The goal of WSD is to ground the meaning of words in certain contexts into concepts as defined in some dictionary or lexical repository. WSD requires two main information sources: context and knowledge. Sense-tagged corpora provide knowledge, leading to data-driven or corpus-based WSD. Use of lexicons or encyclopedia leads to knowledge-driven WSD. Context is established from neighbouring words and the domain of discourse.

While simple approaches like the naive Bayes classifier or Corpus Lesk algorithm, more complex neural network approaches became popular in the 2010s. As an example, Wiriyathammabhum et al. (2012) applied Deep Belief Networks (DBN) and Yuan et al. (2016) proposed a semi-supervised LSTM model with label propagation.

Related work

The English variant of this task is described by Wang et. al. [1] with a Word-in-Context (Wic) task. This task is defined as a binary classification of sentence pairs. Given two sentences who both contain the specified (target) word, the task determines whether the target word is used in the same context (sense) in both sentences.

Dataset used in the mentioned task is also named Wic (ang. *Word-in-Context database*) [2]. It was constructed from example usages in various lexical resources and compiled with constraints of not having more than three instances for some target word and not having repeated contextual sentences across instances. This paper includes a couple of algorithms where results vary. The results are presented with the F-score which combines precision and recall into a single metric. In this report the score obtained was 71.5%.

In the article [3] the corpus is constructed using stop word removal, which can reduce the size of the corpus up to 25% and improve the accuracy of the model. The article provides insights to two different methods of word sense disambiguation - shallow approach where the model only considers the surrounding words and the deep approach where the model is knowledge-based and supervised. The end results of this paper show the F-score is around 80%.

In the paper [4] the approach to word sense disambiguation is tackled with the use of ”Bag of words” and ”Modified Lesk”. The Bag of words model is used to find out the actual meaning o f a word having different meaning due to different contexts. There is a bag for each sense of an observed word and all the bags are manually populated. When disambiguation occurs the entire sentence is separated into single words and stop words are removed. After this removal each word would be compared with each word of each bag (where senses are stored). The algorithm searches for the maximum frequency of words in common. The results for the Bag of words approach in this paper vary based on sample input. On average the F-score is around 66%.

In case we don't have a labeled dataset available, we could perform unsupervised learning. One of the methods is Word sense induction, specifically context clustering. The method works under the assumption that words are contextually similar if they appear in similar windows of context [5].

Corpus construction

At first we extracted Slovene sentences from *ccGigafida* corpus [6]. In the preprocessing step we performed extraction from *XML TEI format* encoding where we took each word of the sentence along with its lemma to construct actual and lemma sentence. We also noted the beginning and end index of the observed word as well as its position in the lemma sentence. We stored both pairs sentences as well as index information in a *json* file.

In order to obtain a list of homonyms, we needed to find a dataset and a way of extraction. A good source of homonyms is *sloWNet* [7], which contains over 70,000 Slovene literals. The literal meaning of a word is its original, basic meaning. Each entry always contains at least one English literal and optional Slovenian literals, as well as optional usages and definitions in both languages. Since the file is in *XML TEI format*, we needed to convert it into a *json* file. We decided to group entries by Slovenian literals, where each new entry has its list of Slovenian and English literals, that appeared alongside of it, as well as usages and definitions in both languages, in case we will need them in the future. To extract entries, we can sort the list by the number of Slovenian literals each entry has, since one could argue, that the more synonyms each word has, the more likely it is to have multiple meanings. While this is not a perfect way of gathering homonyms, since we still need to handpick them from the top list, it simplifies the process, narrows down the search and gives us a much better overview of the word's meanings.

Before we begin with the context extraction, we decided to *simplify, clean up* and remove Slovenian stop words from the corpus. These are commonly used words in a selected language that carry very little useful information. This way we decrease the computational work required to parse them. As stated from the article [3] the removal of stop words reduced the corpus size up to 25% whereas in our case the percentage removed combined with the simplifying of the text is on average around 52% (depends on the selected word). We removed the stop words in the *bag of words* and *lesk* algorithm since they add noise to the features but did not remove them from the *word2vec* algorithm which requires the whole sentences. We obtained the Slovene stop words from the *Github* repository [8]. For each word in the sentence we first convert the word to lower case. Then we check if the current word is or contains a punctuation, number or a special character (here we remove words such as abbreviations). Lastly, we check if the word is present in the list of stop words. If any of the above checks are true, we remove the current word from the sentence. We also noticed it is important to check both the form in which the word is in the sentence and it's lemma form, since some words when converted to its lemma form become stop words (for example: ”najboljši” is in the lemma form ”dober” which is a stop word).

For context extraction we first tried *bag of words* method as a baseline. We stored words that appear in the window around the target word into a context vector and calculated the number of occurrences in a specific sentence. To determine whether the context of the target word in a pair of sentences is the same, we computed cosine similarity between context vectors of each sentence. We used a predetermined boundary of the value of cosine similarity to decide whether the word was used in the same context.

The Second method we tried so far is the simplified Lesk algorithm. We compared lemma forms of words that appeared in sentences with glosses for different meanings taken from Dictionary of Standard Slovene Language (SSKJ). We obtained the entire dictionary together with word glosses in *.txt* file from Github repository https://github.com/van123helsing/SSKJ. We performed preprocessing, where we transformed a gloss into list of lemmas that are contained in the gloss. We used *CLASSLA* pipeline [9] to get lemma forms of words in glosses and stored them in *json* file.

Third and currently last context extraction method we tried so far is *Doc2Vec* which is an extension of *Word2Vec* embedding. We tagged every word from all the extracted sentences and trained a *Doc2Vec* model. Then we looped through pairs of sentences and gave the first sentence of the pair as new sentence to the model which it then computed similarity to other sentences already present in the model. We extracted the computed similarity to the other sentence in the pair and if the computed similarity was above a predefined threshold, the contexts were deemed similar.

Results

We tested each context extraction method by extracting 50 sentences where the context was declared as similar and 50 where it was not. For those 100 sentences we annotated by hand whether the context usage of a word across both sentences was the same. We tested context extraction on 4 known Slovene homonyms: ”list”, ”para”, ”prst” and ”vila”. Results are shown in table 1.

As we can see, simplified Lesk and Doc2Vec produced similar results in all observed metrics. An anomaly is in Bag of words approach where specificity is 1. This occurred because bag of words checks for similar words around the target word and thus successfully recognizes some expressions (e.g. *pokazati s prstom*), so no false

Method	F_1 Score	Specificity	Sensitivity
Bag of words	0.59	1	0.56
Simplified Lesk	0.66	0.68	0.63
Doc2Vec	0.67	0.68	0.62

Table 1. Results on different context extraction methods.

positives were recorded thus specificity was 1. So overall, simplified Lesk and Doc2Vec produced better results than bag of words, which is the expected result.

Next steps

The next step in our project is to try some other word embeddings and sentence representations in order to obtain context in which the target word is used in a sentence. We were thinking of trying *SentenceBERT* which has a architecture to provided 2 sentences as an input which are passed to BERT models to generate embeddings. Then it uses these embeddings to calculate cosine similarity. We will also attempt to perform clustering on these embeddings so we could try *Word sense induction.*

For the analysis we would like to test our dataset and corpus on already trained models with different techniques. We found a *Word Sense Disambiguation* [10] which is based on the paper [11] and uses BERT ELMo and Flair methods.

References

[1] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[2] Mohammad Taher Pilehvar and José Camacho-Collados. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121, 2018.

[3] Madeeh Nayer El-Gedawy. Using fuzzifiers to solve word sense ambiguation in arabic language. *International Journal of Computer Applications*, 2013.

[4] Alok Pal, Anirban Kundu, Abhay Singh, Raj Shekhar, and Kunal Sinha. An approach to word sense disambiguation combining modified lesk and bag-of-words. volume 3, 02 2013.

[5] Mohammad Nasiruddin. A state of the art of word sense induction: A way towards word sense disambiguation for under-resourced languages. *arXiv preprint arXiv:1310.1425*, 2013.

[6] Nataša Logar, Tomaž Erjavec, Simon Krek, Miha Grčar, and Peter Holozan. Written corpus ccGigafida 1.0, 2013. Slovenian language resource repository CLARIN.SI.

[7] Darja Fišer. Semantic lexicon of slovene 3.1, 2015. Slovenian language resource repository CLARIN.SI.

[8] Gene Diaz. Stopwords slovene (sl), 2016. collection of stopwords for the slovene language.

[9] Nikola Ljubešić and Kaja Dobrovoljc. What does neural bring? analysing improvements in morphosyntactic annotation and lemmatisation of Slovenian, Croatian and Serbian. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 29–34, Florence, Italy, August 2019. Association for Computational Linguistics.

[10] Word sense disambiguation using bert, elmo and flair, 2019.

[11] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*, 2019.