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

David Miškić, Kim Ana Badovinac, Sabina Matjašič

Abstract

Words have different meanings based on the context of their usage in the sentence. If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence. Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of the word is activated by the use of the word in a particular context. Lexical ambiguity, syntactic or semantic, is one of the very first problems that any NLP system faces. In this paper, we review a few methods to solve the disambiguation problem. First, we generated data to get a working dataset. We used the LSTM model which needs a large corpus to do well. Method word2vec seems to be less dependent on corpus vocabulary and is way slower than others. We did not get the best results with TF-IDF model, so we used the dictionary method, which uses the definitions of each word from SSKJ, then translates the word into English and performs sense similarity between the words and selects the most similar definition in the sentence.

Keywords

sense, disambiguation, natural language processing, cross-lingual, word in context

Advisors: Slavko Žitnik

Introduction

Many words have multiple meanings, which are determined by their context. We can determine the specific meaning with the help of word-sense disambiguation (WSD). The problem has been long active in the field of natural language processing, as those languages are context-dependent and understanding is not possible without knowing the meaning [1]. The hardness of the problem has been described as equivalent to solving central AI problems [2]. WSD relies on knowledge, but those can vary considerably depending on corpora, and there are different approaches with their own standardization. But the need for automatic knowledge is growing with the amount of unstructured data and the use of machine translation. There are multiple approaches we will explore in the following section, some of those aim to disambiguate a text in one language by exploiting its differences with another language in a parallel corpus. This is cross-lingual word sense disambiguation (CLWSD).

Related Work

Manual encoding

Wilks, 1972

Preference semantics assumed that each sense has a formula that could determine the meaning. The formula was composed of hierarchical selectional constraints, but those could be loosened within some conditions. In addition, verbs and adjectives listed the preferred sense of the words they modified. Disambiguation involved choosing a formula that was satisfied the most.

One example is the use of the verb to drink in sentences ”The adder drank from the pool” and ”My car drinks gasoline”. The verb prefers an animate subject. In the first case, the adder can have the meaning of machine or animate object, so the animate meaning satisfies the verb’s condition. In the second case, the car is not animate so it does not fit the verb. The meaning of the verb has to be extended and a new usage is added. In this way, the meanings can be dynamically evolved over the text [1].

Small 1980

Word expert parsing presumes the lexicon to be the source of information, but the representation is build up from disambiguation as well. Small’s unconfirmed hypothesis was that human knowledge is more word-based rather than rule-based, so each such word would need its own disambiguator [1].

Dictionary based

Lesk 1986

Optimizing dictionary definition overlap with the help of the dictionary, where the sense can be determined depending on the sense of the words close to it, based on dictionary definitions. One example is the word ”pine cone” where pine

is either a tree or a verb, while the cone is either a body or a fruit. The overlap is in the words evergreen and a tree in the dictionary definition, so those meaning were picked (pine as tree and cone as a fruit) [1]. This method was used with an English translation.

Connectionist based

Waltz and Pollack 1985

The approach was to carry out syntactic and semantic processing in parallel, based on psycholinguistics observation. Such an approach attempted to model the errors as well, one example is the sentence "The astronomer married the star" where humans make a temporary mistake of regarding the word star as a celestial body before the more probable sense of celebrity is picked because the semantic effect of the word astronomer is stronger than case selection of the verb to marry. The model is composed of a network of nodes, representing case frames, semantic priming, and syntactic preferences [1].

Cross Lingual

The sense projection (SP) approach can bootstrap the creation of sense-annotated parallel corpora by exploiting existing resources in well-represented languages, with word alignment and connected sense inventories as the only requirements. [3]

Unlike SP, the multilingual sense intersection (SI) approach does not require any of the texts in a parallel corpus to be sense-annotated, so it can be applied to a wider range of existing resources. Its logical foundation is in that a polysemous word in a language is likely to be translated into different words in other languages, so the comparison with the semantic space of each translation help select the sense actually intended. [3]

Dataset description

The selection of starting candidate ambiguous words was made by using a Slovene dictionary of Homonyms [4]. We collected the corpus from the written corpus ccGigafida [5]. This corpus contains only approximately 9% of the Gigafida corpus but still contains many various themed documents. Using a list of known ambiguous words, we collected the sentences including these words from the ccGigafida.

Subsequently, we wanted to get suggestions for more candidates for ambiguous words based on the text we have. From the existing ccGigafida corpus or any other text source that we have chosen, we check every plausible word for ambiguity. We determine if the word is ambiguous by checking its input in the web dictionaries SSKJ [6] or PONS [7]. If the word has multiple entries in the dictionary and its descriptions are uniquely different, that word is suggested as ambiguous.

We tested TF-IDF (term frequency-inverse document frequency) for annotating gathered data since we have enough context surrounding the keywords. We can quantify the similarity between two sentences using cosine similarity. We find the maximum similarity argument of one sentence and all of the others we manually provided. If two sentences contain similar language (tokens) they will be closer together.

Methods

Preprocessing data

After all the corpora were gathered we had to process it so that everything had an uniform form. Tokenization is a common task in NLP. It is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords [8]. The token occurrences in a document can be used directly as a vector representing that document. We used a Slovene tokenizer from classla library in Python and we also removed Slovene stopwords which we downloaded from nltk Python library.

Disambiguation methods

TF-IDF

For feature extraction, we used the TF-IDF method, which is short for term frequency-inverse document frequency. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches for information retrieval, text mining, and user modeling. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general [9]. We used vectorization, where each tokenized sentence in our corpus is vectorized with unigrams and bigrams. The query is transformed into a vector and then compared with vectors of the corpus. To measure similarity between compared vectors we computed cosine distance, and the most similar is chosen.

word2vec

Word2vec is a technique for natural language processing. It is a type of word embedding which is one of the most commonly used representations of document vocabulary. Word embeddings represent individual words as real-valued vectors in lower-dimensional space [10]. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence [11]. In our case, we compared input sentences with sentences in data collection. Then we break sentences down into words. For words in each sentence, we calculate smooth inverse frequency and SIF with similarity being the sum of SIF* vector for each word and its vector. This is how we achieved that all word embeddings can be combined into one. Finally, we compared these sentence embeddings with cosine similarity. If the new word is not in the model, we used translation to English with google translate (so if a word has more senses, it might translate to one) and take the word in the model that is most sense similar based on the Leacock Chodorow method [12].

For implementation, we used pre-trained word vectors, which can be used for 294 languages. The model was trained on Wikipedia using fastText [13].

Neural network

We trained a neural network utilizing Long Short-Term Memory (LSTM) layers [14]. LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. LSTMs are explicitly designed to avoid the long-term dependency problem, which is achieved with a chain structure that contains four neural networks and different memory blocks called cells [15]. To train our model, we used pre-trained embeddings and precomputed word frequencies [16]. We increased the amount of vocabulary by generating similar tokenized sentences with word2vec.

Dictionary based

We used the idea of the Lesk algorithm and SSKJ for the source of dictionary definitions. The difference was using the translation of input sentence and definitions so word sense similarity could be computed in English, where there are more tools available. The algorithm is unsupervised, having no need for pre-classified or training data, but it is important to have sentences with approximately similar vocabulary to definitions.

Classifying untagged instances

GPT-2

GPT-2 is a Transformer-based model trained for language modeling. [17] GPT-2 was trained on 40 GB of high-quality content using the simple task of predicting the next word [18]. It can be fine-tuned to solve a diverse amount of natural language processing (NLP) problems such as text generation, summarization, question answering, translation, and sentiment analysis, among others. Each pre-trained transformers model has its corresponding tokenizer that should be used in order to preserve the same way of transforming words into tokens. It splits text into tokens (words or subwords, punctuation, etc.) and then converts them into numbers. GPT-2 uses Byte-Pair Encoding (BPE) with space tokenization as pre-tokenization. We used sl-gpt2 [19], which is a transformers model pre-trained on a very large corpus of Slovenian data in a self-supervised fashion. This means it was pre-trained on the raw texts only, with no humans labeling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts.

sloBERTa

The presented SloBERTa model is a monolingual Slovene BERT-like model and is closely related to the French Camembert model [20], which uses the same architecture and training approach as the RoBERTa-based model [21], but uses a different tokenization model [22]. The corpora used for training the model have 3.47 billion tokens in total and the subword vocabulary contains 32,000 tokens [23].

Embeddings from Language Models (ELMo)

ELMo is a deep contextualized word representation that models both complex characteristics of word use, and how these uses vary across linguistic contexts [24]. ELMo language model used to produce contextual word embeddings, trained on large monolingual corpora for 7 languages: Slovenian, Croatian, Finnish, Estonian, Latvian, Lithuanian, and Swedish. Each language's model was trained for approximately 10 epochs. Corpora sizes used in training range from over 270 M tokens in Latvian to almost 2 B tokens in Croatian. About 1 million most common tokens were provided as vocabulary during the training for each language model [25].

K-nearest neighbors (KNN)

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised earning algorithm used for both regression and classification, which uses proximity to make classifications or predictions about the grouping of an individual data point. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points that are closest to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and the class that holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points [26]. While it can be used for either regression or classification problems, it is typically used as a classification algorithm. [27].

One-vs-all (OvA)

One-vs-all is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident [28]. This strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency, one advantage of this approach is its interpretability. Since each class is represented by one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice [29].

Experiments

In this section, we go into further detail about how exactly the above methods were used. We conducted several experiments for cross-lingual sense disambiguation. For evaluating performance we calculate the accuracy of every method that we used and compared the results.

The graph below 1shows the implementation workflow of our methods. We first used manual data and obtained generated data using the LSTM algorithm. We then obtained unannotated data using manual data and the clain sentences

method. Unannotated data was combined with manual data and an extended corpus was obtained using a classifier. We then combined the extended corpus with the generated data and got a working dataset. Finally, we used a working dataset for the three disambiguation methods and obtained the results.

Workflow

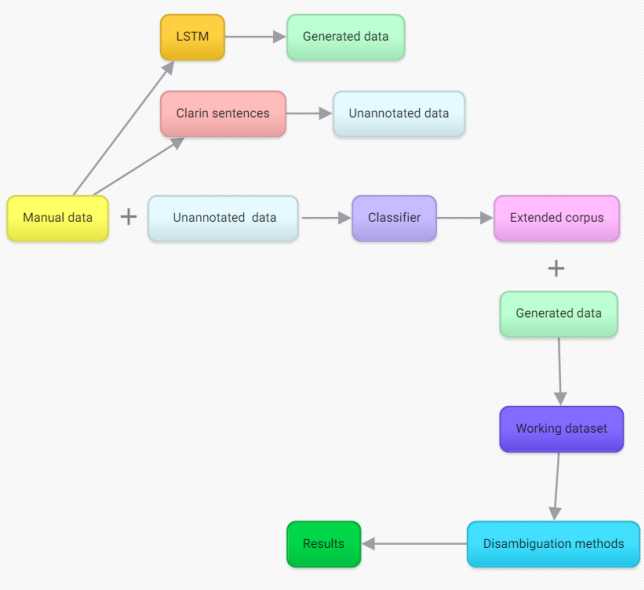

Figure 1. Workflow of our implementation

Disambiguation methods

The TF-IDF method is not the best choice for our problem as we obtained low accuracy, which is 54%. Due to poor results, we added a method that uses the definitions of each word from SSKJ, then translates the word into English and performs sense similarity between the words and selects the most similar definition in the sentence. This method is unsupervised and dependent on exhaustive dictionary definitions. Using the dictionary method, we observed an improvement in the results, we got 61% accuracy. We also implemented the LSTM model. While testing the model, we found that the model needs a large corpus to do well. On a larger corpus, we got 74% accuracy. Lastly, we implemented the word2vec model that works the slowest but is less dependent on corpus vocabulary than other methods.

Text generation

Three methods were used to generate the text. We tried gpt2 first, but after reviewing the results, we found the sentences meaningless. We also tried the sloBERTa and LSTM method, which gave us more meaningful results. Examples are listed below.

1. Sl-gpt2:
    - **Input**: miška
    - **Output**: Miška je bila v svoji mladosti zelo dobra prijateljica z igralcem Bobom Zupanom. Ko je
2. sloBERTa:
    - **Input**: Mačka je lovila <mask>, ki se je skrila v koruzo.
    - **Output**: miš, ribo, žival, miši, hrano
3. LSTM:
    - **Input**: Ne prečkaj te zebre ne da bi pogledal promet.
    - **Output**: cesta umazan zebra videti

Classification

We tried several methods for classification. We implemented the average vector with embeddings from sloBERTa, ELMo, gpt2, and word2vec. We got the worst results with gpt2 embeddings, which is 46% accuracy. With the sloBERTa method, we got 72% accuracy. We used k-nearest neighbors and one vs all. Elmo embeddings worked best, we got 92% accuracy, and kNN was quite faster than one-vs-all.

Discusion

Overall, we learned that some methods show good results for sense disambiguation. When it comes to cross-lingual, used approaches do not work very well. This is probably due to the limited dataset and there are not many quality models that have been trained in the Slovenian language.

Conclusion

In our work, we explore different approaches for cross-lingual sense disambiguation. We went through the whole process from data cleaning, data processing, feature extraction, generating text, using disambiguation methods, and getting the results. We used multiple techniques so that we got a variety of results that we compared. The best embedding method was ELMo with a very high accuracy of 92%. ELMo can uniquely account for a word's context. Other language models such as GloVe, Bag of Words, and Word2Vec simply produce an embedding based on the literal spelling of a word. They do not factor in how the word is being used. This is probably why ELMo method performed so well. The best disambiguation method is LSTM with average accuracy above 74%. This method gave us the best results because LSTM facilitated us to give a sentence as an input for prediction rather than just one word, which is much more convenient in NLP and makes it more efficient [30]. We got the worst results with the gpt2 embedding method and the worst disambiguation method was TF-IDF. This method computes document similarity directly in the word-count space, which may be slow for large vocabulary, and assumes that the counts of different words provide independent evidence of similarity [31].

There are multiple directions for further improving our work. For example, we could try to improve data processing methods and embeddings or by analyzing it more deeply.

There are also other methods for classification we could experiment with. In general, our implementations achieve pretty good results on such a small dataset.

References

[1] *The Oxford handbook of computational linguistics*. Oxford University Press, 2004. Bibliografija ob posameznih poglavjih Kazali.

[2] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), feb 2009.

[3] Francis Bond and Giulia Bonansinga. *Exploring Cross-Lingual Sense Mapping in a Multilingual Parallel Corpus*, pages 56–61. 01 2015.

[4] Júlia Bálint. *Slovar slovenskih homonimov: na podlagi gesel Slovarja slovenskega knjižnega jezika*. Znanstveni Institut Filozofske Fakultete, 1997.

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

[6] Fran/iskanje, April 2022. [Online; accessed 29. Apr. 2022].

[7] Pons slovar, April 2022. [Online; accessed 29. Apr. 2022].

[8] What is Tokenization | Tokenization In NLP, July 2021. [Online; accessed 29. Apr. 2022].

[9] Anand Rajaraman, Jure Leskovec, and Jeffrey D. Ullman. Mining of Massive Datasets. *Mining of Massive Datasets*, January 2014.

[10] A Gentle Introduction to the Bag-of-Words Model, August 2019. [Online; accessed 29. Apr. 2022].

[11] A Beginner's Guide to Word2Vec and Neural Word Embeddings, October 2021. [Online; accessed 29. Apr. 2022].

[12] Claudia Leacock and Martin Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. *WordNet: An Electronic Lexical Database*, 49(2):265—283, January 1998.

[13] 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.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural Comput.*, 9(8):1735–80, December 1997.

[15] Understanding LSTM Networks – colah's blog, January 2022. [Online; accessed 29. Apr. 2022].

[16] jvparidon. subs2vec, April 2022. [Online; accessed 29. Apr. 2022].

[17] How to Build an AI Text Generator: Text Generation with a GPT-2 Model - Oursky Code Blog, February 2021. [Online; accessed 24. May 2022].

[18] Klaudia Nazarko. Word-level text generation using GPT-2, LSTM and Markov Chain | Towards Data Science. *Medium*, December 2021.

[19] macedonizer/sl-gpt2 · Hugging Face, May 2022. [Online; accessed 24. May 2022].

[20] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics.

[21] 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*, July 2019.

[22] Matej Ulčar and Marko Robnik-Šikonja. Slovenian RoBERTa contextual embeddings model: SloBERTa 1.0, 2020. Slovenian language resource repository CLARIN.SI.

[23] EMBEDDIA/sloberta · Hugging Face, May 2022. [Online; accessed 24. May 2022].

[24] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv*, February 2018.

[25] Matej Ulčar. ELMo embeddings models for seven languages, 2019. Slovenian language resource repository CLARIN.SI.

[26] Antony Christopher. K-Nearest Neighbor - The Startup - Medium. *Medium*, December 2021.

[27] What is the k-nearest neighbors algorithm? | IBM, May 2022. [Online; accessed 24. May 2022].

[28] One-vs-Rest and One-vs-One for Multi-Class Classification, April 2021. [Online; accessed 24. May 2022].

[29] sklearn.multiclass.OneVsRestClassifier, May 2022. [Online; accessed 25. May 2022].

[30] LSTM for Text Classification | Beginners Guide to Text Classification, June 2021. [Online; accessed 25. May 2022].

[31] November 2008. [Online; accessed 25. May 2022].