Bachelorarbeit

Debate Topic Expansion

Chieh Kang

21 January 2022

Texttechnologie der Goethe-Universität Frankfurt am Main
Prof. Dr. A. Mehler

Abstract

Given a debate topic, it is often to make an expansion of the topic, the reasons can be the followings: (1) The scope of the debate topic is too shallow and we eager to discuss more. (2) A debate topic is sometimes related to the others and the discussion will not be complete when we do not discuss the others as well. (3) We may want to discuss the particular concept or the core the debate topic. It's thus meaningful to build a model in order to find the expansions of the topics.
IBM® Research Team has proposed a method to expand the boundary and find the expansion topics of the given debate topics in 2019. There are two types of topic expansions in their paper, consistent and contrastive expansions. We focus on the consistent expansions. Consistent expansions are defined as the expansions that expand our topics in a positive way or at least neutral.
The main objective of this paper is to follow and examine the steps of IBM® Research Team's idea and since the original discusses the model in english, we would like to implement a topic expansion model with 7 steps, including pattern extraction, filtering, training, etc, in another language (german) using translator and compare the result between different models to propose the final german model at the end.

Contents

*

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Can AI or robot communicate with people? This has always been a question that haunts in our mind. The first talking chatbot is ELIZA (Weizenbaum 1966), developed at the MIT Artificial Intelligence Laboratory by Joseph Weizenbaum. And IBM® 's *Watson*[1], which plays against human contestants live on Jeopardy![2] in 2011, is able to answer questions without only using a fixed template. In addition, intelligent personal assistants like Siri® have also been more and more popular in recent years, To develop an AI that can form complex arguments is not easy. To achieve the goal, many different NLP tasks are often included in the model. From the simple tasks like morphological analysis, which aims to analyze the word form and the lemma, to more difficult tasks like speech recognition, that builds a model to recognize voice, spoken language and transform them into sentences and texts, or natural-language understanding (NLU), which seeks to let computer understand human languages.

1.2. Project Debater

IBM®'s *project debater*, which was introduced in 2019 and aims to help people form better arguments and make better decisions on the problems that are not only black-and-white, brings it to a new level. It is the first AI system that can debate on complex topics. And it has also held several live debates with human debate champions.[3] There is some room for improvement but still quite astonishing. And to develop such an AI system, one of the very first and important steps of *project debater* is topic expansion. Given a debate topic, it's important to first define and then to extend content of the debate under the former definition. For example, given the debate topic: Should same-sex marriage be legalized?, we would probably want to discuss a little bit further. For the same-sex marriage supporters, they may say that same-sex marriage is a "human right" and for the opponents, they may refer to the "religion". In this way, the whole debate boundaries are being extended. "Human right" and "religion" are thus our expansion topics in this case. The goal of topic expansion

[1]Registered trademark of IBM Corp.
[2]Registered trademark of Jeopardy Productions Inc
[3]The first live debate can be seen at `https://www.youtube.com/watch?v=3_yy0dnIc58`

is to find these expanded topics from a given topic.
IBM® Research Team has detailedly describe this task (Bar-Haim et al. 2019), including presenting algorithms for finding consistent and contrastive expansions and training a model that can identify good expansions. Consistent expansions are the expansions that expand our topics in a positive way or at least neutral, for example ”Tesla” is a consistent topic of ”Electric car”. Consistent expansions help us to go deeper into the definitions or applications of our given topics. On the other hand, contrastive expansions are the expansions that are mostly the antonyms or the opposite opinions of given topics. For example, ”Dictatorship” is the contrastive expansion of ”Democracy”. In our paper, we will focus on finding the consistent expansions.

1.3. Task

We imply this process also for the german language and compare the results. There are two main tasks in this paper. First, building a model to extract and train the expansions following some of the steps from (Bar-Haim et al. 2019). The first step is building a corpus, translation(english to german) in german model, and extracting the EC (expansion concept) given DC (debate concept) and patterns. DC is a debate topic we wish to find the expansion for and EC is a debate topic that is an expansion from DC. The second step includes filtering and input features extraction. In the last step, all the topic pairs that are not filtered out are annotated as good or bad expansions manually and fed into a logistic regression model to do the training. The parameters are adjusted to better meet our demand of the model. Second, doing german translation und use the model on topics extraction and training on german language. The performance of different parameters, different word embeddings, and different input features for the training as well are being explored to compare the result between different models.

1.4. Overview of the paper

Chapter 2 gives a short introduction to all of the background knowledges that are crucial to better under the model and experiments. Chapter 3 describes all of the steps how the model is built. From how the corpus is built to how the model is trained. In Chapter 4 several experiments are done in order to find out which factors could effect the result and also the final version of model is proposed. Chapter 5 gives the short summary and also the possible improvements in the future.

2. Background

2.1. Dependency parsing

Dependency parsing is a term in linguistic. which aims to analyze structure of a sentence. Unlike other methods, the word order in the sentence does not really matter but only the relations between words. An example can be seen in image 2.1. The word above the arrow is the relations from A to B. For example, ”chocolate” is a dobj of ”eat”. Dobj stands for direct object, which is a noun phrase of the verb.

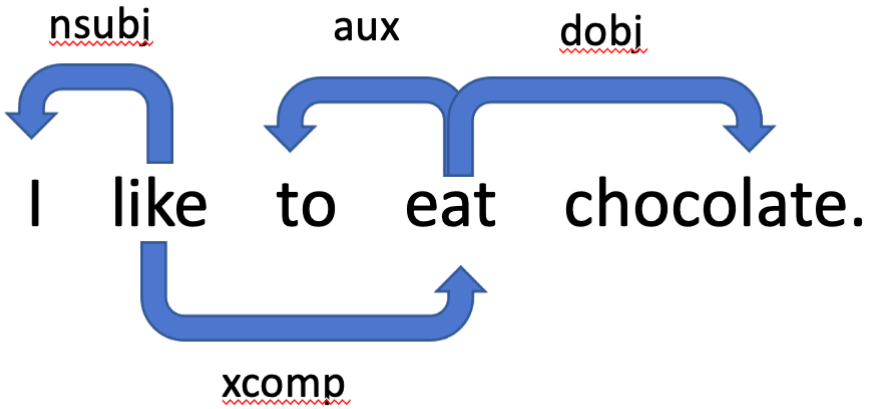


Figure 2.1.: Dependency relations.

2.2. Word embedding

In Natural language processing, the words, phrases, sentences or even texts are often represented as a real-valued vector. Assume that we have $n \in \mathbb{N}$ words in total in our corpus, we want to map these n words to a much lower dimensional vector space. One of the reasons why it's good to map into a much lower dimensional vector space instead of staying in a high dimensional vector space is that it's easier to compute, saving computing power and also

the words that are related can be grouped together. The words with similar meaning are normally closer in the vector space.

2.2.1. Pre-trained model

Pre-trained word embeddings, also called static word embeddings are the word embeddings that are already trained before in another task or anther corpus in order to save the training time and thus can be used in another NLP task again. One of the benefits is that it's easier to find a word representation for rare words. It's not uncommon that a word in our task happens to be a really rare word. It is not easy to find an appropriate word representation for them. One of the solutions is pre-trained embeddings. Training on a really big dataset that contains a lot of rare words beforehand, and then extract the word embeddings when we need them.

2.2.1.1. Word2vec

One of the pre-trained models os Word2vec. In 2013, a team of Google published word2vec (Mikolov, Sutskever, et al. 2013). They focus on the word representations learned by neural network. Word2vec has two different model architectures: CBOW and Skip-gram. CBOW predicts current based on the other words in the context and the word order does not matter at all, on the other hand Skip-gram predicts the surroundings words based on the current word. With the model architecture word2vec can be trained on a few hundred of millions of words, with a modest dimensionality of the word vectors between 50 - 100. And like mentioned in section 2.2, similar words will not only be closer in but also have multiple degrees of similarity. Another somewhat surprising is that these vector-space word embeddings can not only catch the literal meaning but also the hidden relations between two words. An example is that the male/female relation is also been learned in the process (Mikolov, Yih, and Zweig 2013). When we subtract the word embedding of "Man" from the word embedding of "King" and then plus the word embedding of "Woman", we will get a really close representation of "Queen".

2.2.1.2. Fasttext

Fasttext (Bojanowski et al. 2016) is a library for efficient learning of word representations and sentence classification. It is an extension model of Word2vec described in subsubsection 2.2.1.1. Specially, each word in Fasttext is represented as a bag of character character. An example given in paper (Bojanowski et al. 2016) is that the word "where" is being broken into <wh, whe, her, ere, re> when it is n-gram. The benefit of the approach is that the model can better understand the subwords in a relative long word and also the suffixes and prefixes.

2.2.1.3. Shortcoming of pre-trained model

One of the biggest shortcomings of the pre-trained model is that it does not take the word context into consideration. It assumes that a word has the same or at least meaning in

different sentences, which is obviously not the case. For example, for the word 'bank", it has different meanings, "Bank" can mean a place where people borrow money from or keep money in. "Bank" also has the meaning like "river bank". It is thus not appropriate that assign them the same word embedding in different context. Another solution is needed.

2.2.2. Contextualized Word Embeddings

Unlike traditional word embeddings, contextualized word embeddings focus not only the word-level embedding but also take the whole sentence into consideration. Hence, the contextualized word embeddings is not static. The word embedding is generated based on the context of the word in a sentence.

2.2.2.1. BERT

BERT (Devlin et al. 2019) stands for Bidirectional Encoder Representations from Transformer (Vaswani et al. 2017). BERT is a pre-training model which aims to improve the pre-trained language representations. Like the name tells, BERT is the only encode part of Transformer, and also it is bidirectional. That means the model reads the input sequences not only from left to right but also from right to left and thus it provides more choices of the pre-training architectures. Because of these features, it enables BERT to add an extra layer at the top for the downstream tasks using the pre-trained model.

2.2.3. Statified Contextualized Word Embeddings

As mentioned in subsubsection 2.2.1.3, the main shortcoming of static word embeddings is that it does not consider about the word context, hence it outputs poor result when a word has multiple meanings. However, static word embeddings also have some advantages over contextualized word embeddings. First, contextualized word embeddings are computationally expensive, the training takes a lot of time, computing power, memory, etc. On the other hand, the computational cost of static word embeddings is much lower. Second, many NLP tasks still use static word embeddings.

2.2.3.1. X2Static Model

X2Static model (Gupta and Jaggi 2021) is a model which intends to combine the static word embeddings and contextualized word embeddings. They purposed a new way to distill existing contextual word representation models into static word embeddings. First, after data preprocessing, the data are trained on transformer-based models to get the word embeddings. The transformer-based models include: BERT, GPT-2 (Radford et al. 2019), and RoBERTa (Liu et al. 2019). And all the word embeddings are extracted from the last layer of the models. The result is being evaluated on several tasks, including movie reviews, opinion polarity, product reviews, etc. The performance is pretty decent given Fasttext and Glove (Pennington, Socher, and Manning 2014) as the baselines on most of the tasks.

2.3. Semantic Relation

Semantic relation at word level describes the relation between words. There are two main categories that are discussed in the paper: The relations of hypernym, hyponym and co-hyponym could be found in image 2.2

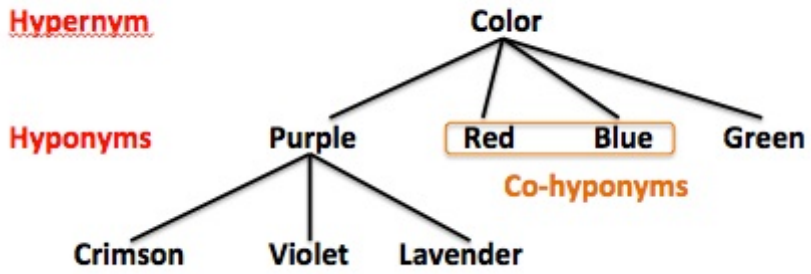


Figure 2.2.: Hypernym/hyponym (and co-hyponym) relation. photograph from Wikipedia user Tanzx30 2017, p. 6

- hypernym: If a word X is a hypernym of another word Y, it means that the meaning of Y is included in the meaning of X. For example, ”color” is a hypernym of ”red” and ”blue”.
- hyponym: If a word X is a hyponym of another word Y, it means that the meaning of X is included in the meaning of Y. For example, ”tiger” and ”lion” are hyponyms of ”animal”.
- co-hyponym: If a word X is a co-hyponym of another word Y, it means that X and Y are both the hyponyms of another word. For example, ”red” and ”green” are co-hyponyms.
- synonym: If a word X is a synonym of another word Y, it means that X has the same meaning or nearly the same meaning of X. For example, ”car” and ”automobile” are synonyms.

2.3.1. Wordnet

WordNet® (University 2010, p. 5) is a large lexical database of English developed by the Cognitive Science Laboratory of Princeton University in 1985. Synsets are used in WordNet® to look up a word (nouns, adjective, verbs, etc.). A synset is a group of synonyms that are interchangeable in some context. For example, ”motorcar” and ”automobile” has both the meaning ”car”, so they both have the synset ”’car.n.01”. WordNet has currently about 117 000 synsets.

2.3.2. Germanet

Germanet (Hamp and Feldweg. 1997, p. 3; Henrich and Hinrichs. 2010, p. 4) is a semantic network, which has much similarity with Wordnet®. Like WordNet®, Germanet also use synsets in its main structure. Germanet also provides the semantic relations like synonyms, hypernyms, antonyms in the database.

2.4. Sentiment Analysis

Sentiment analysis is a processing of extracting and determining the subjective information from an article or a piece of text. Sentiment analysis is widely used to measure the author's attitude as positive, negative or neutral towards a specific topic, for example we can collect all the movie reviews of a movie and train a model to determine whether a certain movie review is positive, negative, or neutral. It helps the movie company to better grasp the thoughts of the public.

2.5. Classification problem

A classification problem is when the output of the model is not continuous but grouped into different classes. For example, a dog image recognization model will classify the output into two groups: dogs and non-dogs. The output of a binary classification model is categorized into two classes and often labeled as either "0" or "1". On the other hand, the output of a multiclass classification model is categorized into more than two classes. In our paper, we focus on binary classification model and two machine training methods: logistic regression and decision tree.

2.5.1. Model

2.5.1.1. Logistic Regression

In the field of machine learning logistic regression is a model that solve the classification tasks linearly. Logistic regression is initially for the binary classification, but it can be extended to multiclass classification. Unlike linear regression, which does not necessarily have to restrict its output domain, the output of logistic regression is between 0 and 1 (just like probability). To achieve it, the logistic function has to be well selected. The sigmoid function is a function which maps any real value to (0,1). Mathematically:

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

And the probability of a training data point x is defined as:

$$\hat{p} = \sigma(\theta^T x)$$

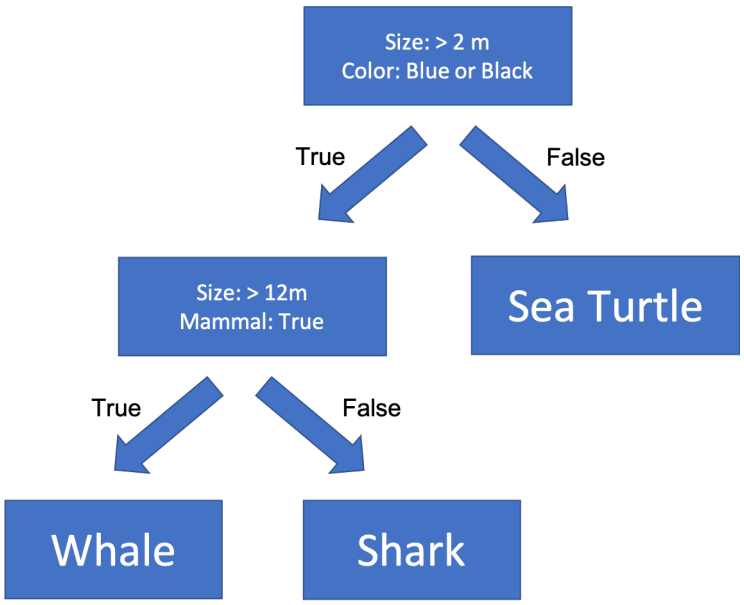


Figure 2.3.: Classifying sea turtles, sharks and whales.

where θ is the vectorized parameters.
At the end, the prediction can be written as:

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{p} >= n \\ 0, & \text{if } \hat{p} < n \end{cases} \tag{2.1}$$

where n is the threshold we can control and can be any real value between 0 and 1.

2.5.1.2. Decision Trees

Decision trees are a machine learning algorithm that can be used for classification. The model is mostly a binary tree structure and based on a sequence of binary decisions. A tree consists of three main parts: internal node, edge and leaf. They represent the test conditions, the outcomes based on previous test conditions and the class labels in a decision tree respectively. A simple example of classifying sea turtles, sharks and whales are shown in image 2.3.

We use both logistic regression and decision trees to do the classification and compare their result.

2.5.2. Performance Measures

A common way to measure the performance is accuracy, the number of times the model predicts correctly divided by the number of times the model predicts in total. But accuracy is sometimes not a great metric when dealing with the imbalanced classification dataset. Precision, recall, or f1-score may be better choices.

2.5.2.1. Cross-validation

Cross-validation is an evaluation method of machine learning. A traditional evaluation method splits the dataset into training dataset and test dataset (sometimes validation dataset too) once and then measure the performance. A k-fold cross-validation splits the dataset into k subsets at the beginning. After that, cross-validation takes k-1 subsets as training data and 1 subset as test data at a time. The chosen test subset are different every time, so the whole process runs k times in total. The advantages of cross-validation are to reduce over-fitting, use all the data, and fine-tune the hyperparameters. In our model, we use a common 10-fold cross-validation as our evaluation method.

2.5.2.2. Precision, recall

To better understand the meaning of precision of recall, let's take an example:
Supposed that the mission of our classifier is to recognize dogs. There are four possible outcomes:

1. A dog is correctly being identified as a dog (True Positive).
2. A non-dog is incorrectly being identified as a dog (False Positive).
3. A dog is incorrectly being identified as a non-dog (False Negative).
4. A non-dog is correctly being identified as a non-dog (True Negative).

If we write them in a table:

Actual \ Predicted	dog	non-dog
dog	True Positive	False Negative
non-dog	False Positive	True Negative

Table 2.1.: Possible outcomes of recognizing dogs

With the help of the above example we can better understand the definition of precision and recall.

1. Precision is defined as the ratio of positive predicted instances that are indeed positive. It is widely used on measuring the case which can not afford the cost of False Positive like detecting email spam. If an important email is being incorrectly classified as spam and thus not read, the cost may be huge. In this case, precision is a good metric. Mathematically:

$$\text{precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

2. Recall is defined as the ratio of positive actual instances that are being predicted correctly. It is widely used on measuring the case which can not afford the cost of False Negative like detecting contagious disease. If a person, who has contagious disease, is tested negative, then it's dangerous for other people. Mathematically:

$$\text{recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

2.5.2.3. Precision-recall tradeoff

Normally, it is desirable to find a model with both high precision and recall. Unfortunately, it is often not possible. Mentioned in equation 2.1, we can adjust thresholds and thus an instance could be categorized differently. Most models tend to increase precision and reduce recall with higher threshold and increase recall and reduce precision with lower threshold.

2.5.2.4. f1-score

Precision and recall are suitable in different situations. But in the situations when precision and recall are both important and thus we need to find a balance of them, f1-score is another good measure. F1-score is defined as:

$$\text{f1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.2}$$

As we see in the equation 2.2, the function takes precision and recall as inputs. When both precision and recall are high and closed to 1, the f1-score is closed to 1 (best score). On the other hand, when both precision and recall are low and closed to 0, the f1-score is closed to 0 (worst score). F1-score will be our main performance measure. There are two reasons: (1) A performance measure based purely on accuracy is not appropriate, since our dataset is imbalanced. (2) Our model does not emphasize False Negative or False Positive, that is to say, the model does focus only on precision or recall, but rather than the balance of them both.

3. Model

3.1. Procedure

There are seven steps of the whole procedure in total, as shown in image 3.1 [1]

1. Selecting Topics
2. Building Corpus
3. Translation
4. Pattern Extraction
5. Filter
6. Input Features
7. Training

3.2. Selecting Topics

First, 50 common debate topics (english) are selected manually. These 50 topics are also article titles of Wikipedia. The list of english topics can be found at appendix A. These topics are then being translated into german. Since not every topic can be translated one by one to german, it is possible that an english topic has more than one german translation. Therefore, there are more german topics, around 72 topics. The lists of german topics can also be found at appendix B.

3.3. Building Corpus

After selecting 50 debate topics in english, the next step is to start to build a corpus related to the 50 debate topics. Since translating the articles to another language takes a lot of time,

[1] The source code can be found at https://github.com/jackykang061233/IBM-Project-Debater-German/tree/master

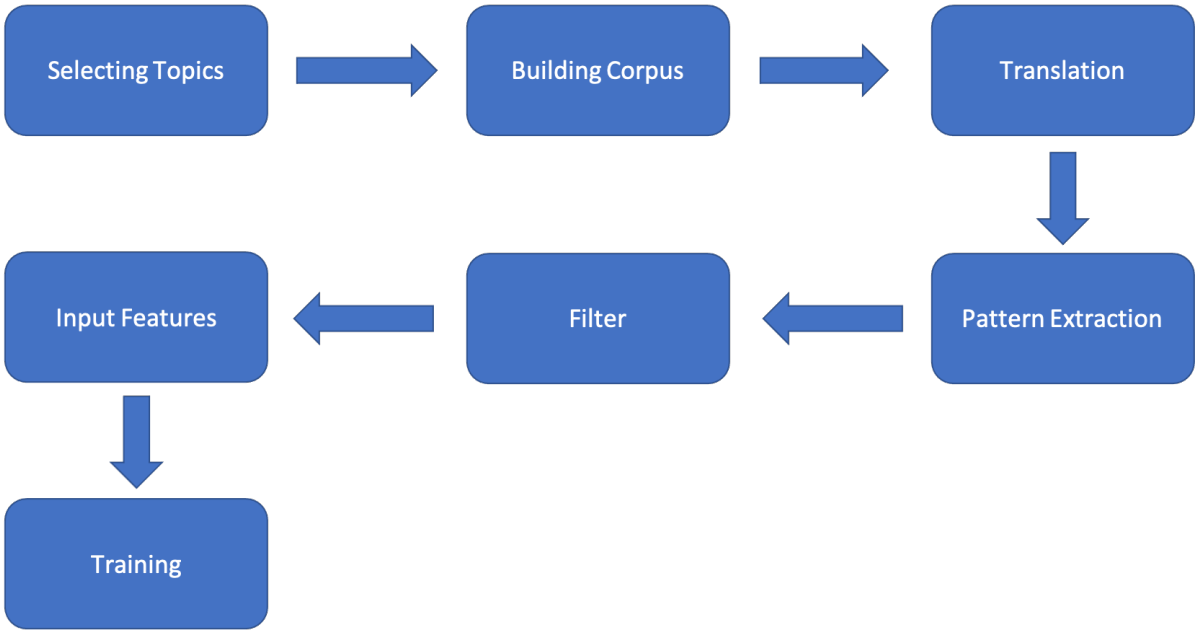


Figure 3.1.: Procedure.

it is impossible to build a corpus consisting of the whole english wikipedia articles, we build the corpus in another way. As mentioned above, these 50 topics are also article titles of Wikipedia. So the corpus is built on the wiki pages of the topics and the wiki pages of their outlinks. For example, the topic ”Gun Control” has the wiki outlink ”Gun laws in Australia”, hence the wiki page of ”Gun laws in Australia” is also being added to the corpus. There are around 26,000 wiki pages, 80,000,000 words and 520,000,000 characters in the corpus.

3.4. Translation

The corpus is then being translated into german using LibreTranslate . LibreTranslate is a free and open source machine translation API and web-app built on top of Argos Translate (Finlay n.d.). Because the LibreTranslate API does not allow paragraph with more than 40000-50000, the corpus is then split into 25000 characters a group to do the translation.

3.5. Pattern Extraction

11 different patterns are being chosen for the task. Sentences containing the wished patterns and debate topics are first being extracted using Regular expression. And then the expansion topics are being extracted with help of Stanza (Qi et al. 2020, p. 1). The list of the patterns can also be found at appendix C. Dependency parser, which is mentioned in section 2.1, is used to extract the expansion topics. The steps of extracting expansion topics:

1. Combine DC and the pattern
2. Label the last word of the pattern as A.
3. Find the head of A, labeled as B.
4. Check if the next word of B is a genitive case.
5. If a genitive case exists, then find the head of the genitive case, labeled as C and extract the words between A and C (included).
6. If a genitive case does not exist, then extract the words between A and B (included).
7. The extracted words are our expansion topics.

A simple example for extracting an expansion topic:

DC: Die Schokolade (English: chocolate)
Sentence: Die Schokolade ist ein Weihnachtsgeschenk meines Vaters.
(English: The chocolate is a Christmas present from my father)
Universal Dependencies: see figure 3.2
Pattern: DC ist ein EC
Steps:

1. Combine ”Die Schokolade” and ”ist ein” as the desired pattern.
2. ”ein” is the last word of the pattern.
3. The head of ”ein” is ”Weihnachtsgeschenk”.
4. The next word of ”Weihnachtsgeschenk” is a genitive case.
5. The head of the genitive case is ”Vaters”.
6. All the words between ”ein” and ”Vater” are extracted as expansion topic (”ein” not included)
7. The expansion topic is Weihnachtsgeschenk meines Vaters.

3.6. Filter

The following step after pattern extraction is filtering. At this step the extracted topics that do not meet the standard are filtered out. There are four main filters for the model:

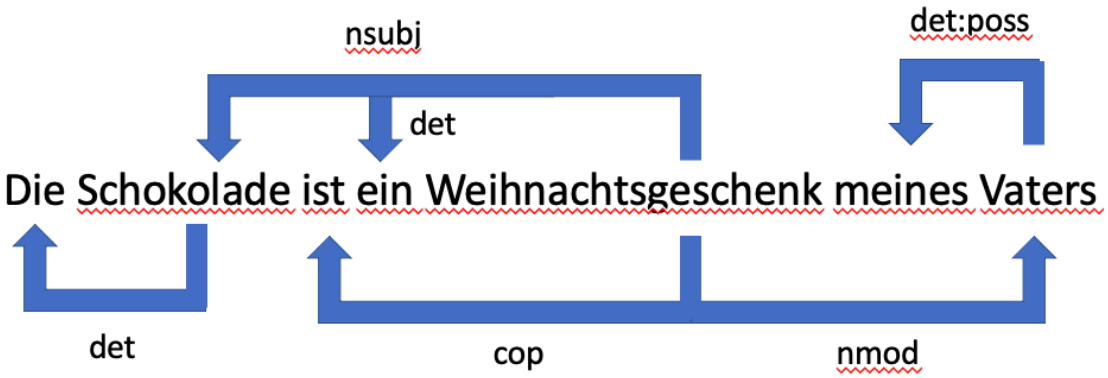


Figure 3.2.: Example sentence for extracting expansion topic.

3.6.1. Stopwords

A stopword is a frequently used word that is discarded during natural language processing in order to save space or to speed up the searching. For example, ”is”, ”will” are stopwords in english and ”oder”, ”dann” are stopwords in german. Hence, if an expansion topic is a stopword, it may not be a good expansion. We require that our expansion topics can not be a stopword. The expansion topics that are stopwords will not enter next step. The english model uses NLTK english stopwords and the german model uses NLTKBird, Klein, and Loper 2009, p. 2 german stopwords. The english model has 179 english stopwords and the german model has 232 german stopwords in total. The reason to choose stopwords as one of the filters is because that stopwords are most frequently used words in a language and thus they are unlikely to contain really specific and useful information that we need.

3.6.2. Frequency Ratio

Frequency of a word is the times a word occur in the corpus. The large divergence of the frequencies between two topics may indicate that the one with low frequency is only a special case, hence might not be a good expansion. We require that frequency ratio should be bigger than 0.01. The idea behind using frequency ratio as one of the filters is that if the discrepancy of the frequency of two topics are really high, that probably means that one topic is only a special case of the other concepts and thus not a good expansion generally.

3.6.3. Substring

A substring is a string in a longer string. For example, ”government” is a substring of ”government of Iraq”. DC can not be a substring of EC and EC can’t be a substring of DC as well. The reason of choosing substring as a filter is that the meaning of a substring overlaps the meaning of the longer string and thus does not really expand the scope of discussion.

3.6.4. Cosine Similarity

Every topic is represented as a word vector. At this step, we first get the word embedding using the method described in 2.2 for every topic and then we calculate the cosine similarity between DC and EC. There are two different kinds of word embeddings we use in this paper: The first one is traditional embedding, in our case Fasttext, which is mentioned in subsubsection 2.2.1.2. We use the Word vectors for 157 languages version (Grave et al. 2018) for english and german. The second one is the statified contextualized word embedding, which is also mentioned in subsection 2.2.3. Fort the english embedding, we use BERT2STATIC$_{sent}$ [2] model from (Gupta and Jaggi 2021). And for the german embedding, we train a german embedding model by ourselves using BERT. The training details are described in 3.6.4.2.

3.6.4.1. Cosine Similarity

Calculating the inner product between two vectors helps us to better understand the angle between two vectors, since for vector v and vector av ($a \in \mathbb{R}^+$), their cosine value is 1 and for vectors pointing to the opposite direction, their cosine value is -1. The pairs with cosine similarity smaller than 0.25 are filtered out.

3.6.4.2. German word embedding

To train a german statified contextualized embedding model, we refer to the BERT2STATIC$_{sent}$ and Textimager (Hemati, Uslu, and Mehler 2016, Hemati 2020). The training steps are as follow:

1. Wikipedia dump files for the german wiki are dowloaded from Wikimedia. [3] The version we choose is from 20190201. All of the articles in german language except for few articles which are unable to extract at the next step are included in this corpus.

2. The whole content of every article in the dump files are extracted with WikiDragon (Gleim, Mehler, and Song 2018). WikiDragon is a tool to extract and clean the articles from Wikipedia dump files.

3. For the next step, use spaCy (Honnibal et al. 2020) is used for the NLP-Preprocessing, which extracted linguistic features (Pos Tagging, Morphology, Lemmatization, Dependency Parsing, etc.) supported for the German language. At the end every article was saved in xmi format.

4. The last step before training the lemma-form of every word in a sentence is extracted from the xmi files and saved in a txt-file for the training.

5. For the training we follow the structure of the BERT2STATIC$_{sent}$ model. The detailed settings of hyperparameters are in

[2] There are different models could be found at https://zenodo.org/record/5055755
[3] The internet address is https://dumps.wikimedia.org/

Epoch	Gradient Descent Algorithm	Learning Rate	Minimum Word Count	Embeddings Size
1	SparseAdam	0.001	10	768

Table 3.1.: Table to hyperparameters of german embedding training

6. The model training uses a single GPU Quadro RTX 8000. The extraction of the lemma-form from Wikipedia dump takes 24 hours and the whole training process takes approximately 50-60 hours.

3.7. Input Features

3.7.1. Cosine Similarity

The cosine similarity mentioned in subsection 3.6.4 is an input feature as well.

3.7.2. Semantic Relation

For the english model we use Wordnet mentioned in subsection 2.3.1 and for the german model we use Germanet mentioned in subsection 2.3.2.

3.7.3. Sentiment Analysis

3.7.3.1. English model

For the english model we use the sentiment-roberta-large-english (Heitmann et al. 2020) model, which is accessible on Hugging Face (Wolf et al. 2020). The model is is a fine-tuned checkpoint of RoBERTa-large and assigns every word or sentence two values: positive(1) and negative(0). The model was fine-tuned and evaluated on many different kinds of text sources and thus outperforms many one-source model.

3.7.3.2. German model

For the german model we select the german-sentiment-bert (Guhr et al. 2020) model, which is also accessible on Hugging Face. The model is BERT-based and trained on more than 1.8 million German-language samples. The domain of training data includes german twitter, movie reviews, microblogs, news, etc. They compare the model trained on Fasttext and BERT, where the latter outperforms the former.

3.7.4. Topic Occurrences

Discussed in subsection 3.6.2. The occurrences of a topic is also an input feature.

3.8. Training

3.8.1. Labeling

Each pair after filtering is annotated either ”1” or ”0” manually. ”1” means good expansion and ”0” means bad expansion.

3.8.2. Training

The sklearn (Pedregosa et al. 2011, Buitinck et al. 2013) models for logistic regression and decision tree are used for the training. All the trainings use the following steps:

1. Preprocessing: Only the needed features remain, the rest is deleted and the target is extracted.
2. Split train test data: The data are split into two groups: training and testing. The split ratio is 0.1 and we use stratified train test split to make sure both ’1’ and ’0’ exist in the dataset.
3. Grid search cross validation: We then use the grid search cross validation to fine-tune the model, for both logistic regression and decision tree. The cross validation is a 5-fold cross validation.
4. The above two steps are repeated 150 times in order to get the broad understanding of the model and the data.
5. Take the average of the 150 f1-scores to get the final f1-score. The f1-score calculated in our model is macro-f1-score.

4. Experiments

4.1. Translate first vs Extraction first

4.1.1. Objective of the experiment

We would like to know which of the following methods has a better performance:

- Translate the english corpus into german then extract the expansion topics using german patterns
- Extract the expansion topics using english patterns then translate the extracted expansion topics into german

There are two reasons for doing this experiment:

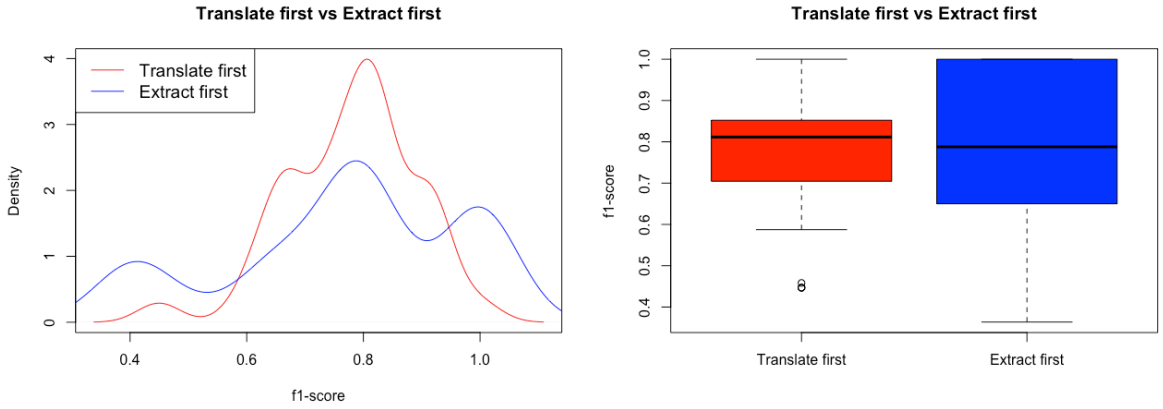
1. If ”Extraction first” has better performance then there is no need to translate the whole corpus and thus less computational cost

4.1.2. How the experiment is done

The only differences between ”Translate first ”and ”Extraction first” are the order of Pattern Extraction and Translation. In ’Translate first”, we will first do the translation then the extraction. And in ’Extract first” we do the opposite, first extract the expansion topics then translate them with help of Libretranslate.

4.1.3. Result

The result, as shown in image 4.1, is a little bit tricky. The size of two samples vary, since ”Translate first” has more than 250 pairs, whereas ”Extract first” has only a little bit more than 60 pairs. So we can see the result from ”Extract first” differ a lot, it has a high variance. The big gap of difference between the two models is due to the similarity. ”Extract first” model filters out about almost 200 more pairs than ”Translate first. We assume the cause for this result is that for ”Translation first” we translate the whole sentences and paragraphs and thus give more information, but for ”Extraction first” we only translate the topic, so the translation quality can sometimes be poor. In this case, we will take the ”Translation first” model.



(a) Translate first vs Extraction first density

(b) Translate first vs Extraction first boxplot

Figure 4.1.: F1-score of Translate first vs Extraction first

4.2. Fasttext vs Statified contextualized word embeddings

4.2.1. Objective of the experiment

We would like to find out if the types of word embedding effect the performance. We compare the result of fasttext and self-trained embedding. Normally the statified contextualized word embedding should output better result. We would like to know if this is also the case in our data

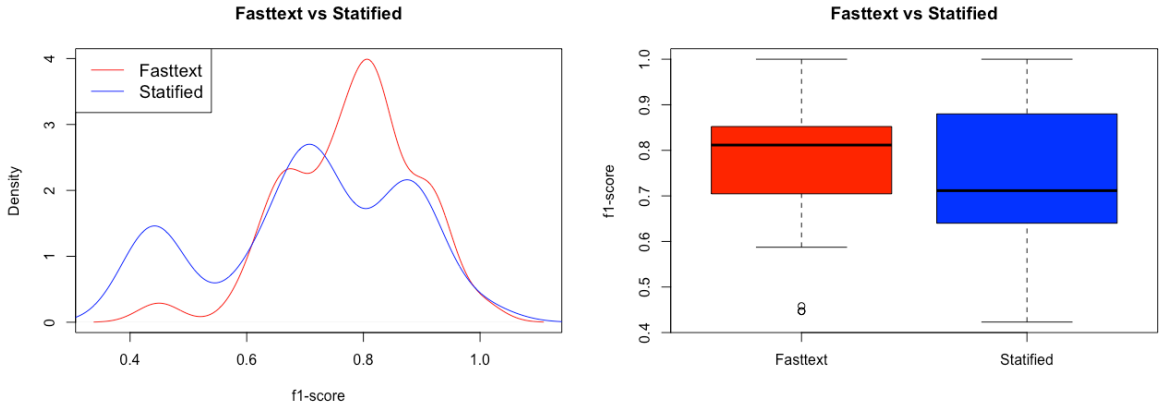
4.2.2. How the experiment is done

For the fasttext, we download the word embedding from the website 'Word vectors for 157 languages' [1]. We first download the 300-dimensional german word embedding then adapt the dimension to 100. For the statified contextualized word embedding, the method is described in 3.6.4.2. The word embedding is the only variable, therefore only the cosine similarity is different. The rest stays the same.

4.2.3. Result

As shown in image 4.2, the average f1-score of fasttext is 0.7785870 and statified contextualized word embeddings 0.7083629. Also the p-value of t-test of the null hypothesis "Fasttext > Statified contextualized word embeddings" is 1, which means that the test is not statistically significant and we can't reject the null hypothesis. Normally, statified contextualized word embeddingsIng should contain more information, but in our case, some of the words do not

[1] https://fasttext.cc/docs/en/crawl-vectors.html



(a) Fasttext vs Statified word embedding density (b) Fasttext vs Statified word embedding boxplot

Figure 4.2.: F1-score of Fasttext vs Statified contextualized word embedding

have a word embedding, so they have to be set to 0 or abandoned, whereas fasttext can find a way to assign the word embedding. We assume this is the reason why fasttext has a better performance.

4.3. Logistic regression vs Decision trees (choosing final model)

4.3.1. Objective of the experiment

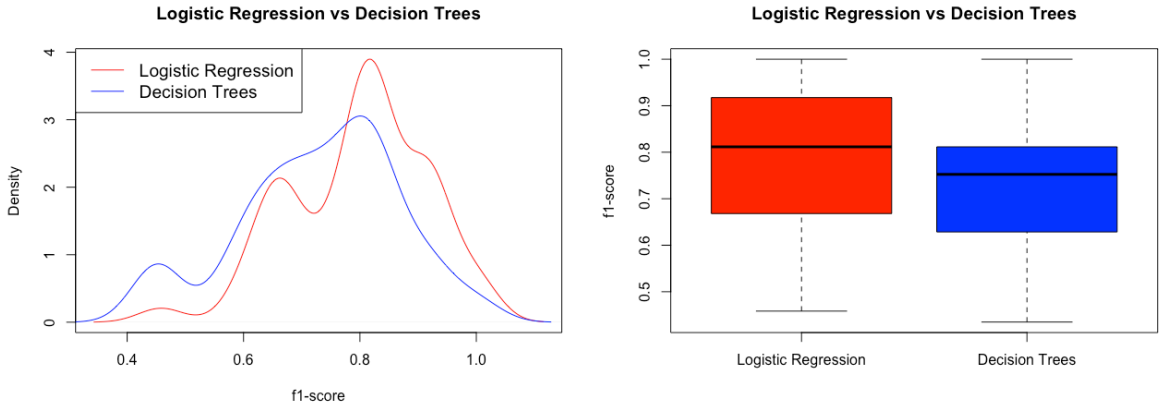
We would like to determine which of machine learning methods fit our data better and has better performance on the classification problem. Logistic regression and decision trees are chosen for the task.
The reasons for selecting these two models:

1. The ways they separate data are different. Logistic regression is a linear classifier, whereas decision trees are a non-linear classifier

2. Decision trees have normally a higher cost than logistic regression.

4.3.2. How the experiment is done

The steps of the two methods before training are the same and already well described in chapter 3. Only the training models in the training phase differ, we also use grid search to find the best hyperparameters for the models.



(a) Logistic regression vs Decision tree density (b) Logistic regression vs Decision tree boxplot

Figure 4.3.: F1-score of Logistic regression vs Decision tree

4.3.3. Result

The comparison between logistic regression and decision trees is shown in image 4.3. The average f1-score of logistic regression is 0.7974715, whereas decision trees have an average f1-score of 0.7310355. We will take the logistic regression model for the training model.

4.4. Final model

From the above three experiments, we implement the ”Translate first”, ’Fasttext” and ”Logistic regression” in our model.

4.4.1. After filtering

There are 255 pairs left from 750 pairs after the filtering. And only 40 of them (about 15%) are good expansions, so our dataset is an imbalanced dataset.

4.4.2. Fine-tuning the model

After fine-tuning the model, we find out that the logistic regression works the best when C is equal to 100 and max iterations are equal to 1000.

4.4.3. Result

The final f1-score is 0.7974715.

5. Conclusion and future work

5.1. Conclusion

In this paper we build a 7-steps german model for the task of topic expansion using translation. Besides the parameters like the ratio of frequency or cosine similaritywe in the filter section, we do three more experiments to find out whether some of the factors also affect the performance. And we get that "Translate first", word embedding using 'Fasttext" and training mode using "Logistic regression" are the better fit for the model. The final model outputs a 0.7974715 f1-score.
Although the model yields a decent f1-score, there are still a few things that can be improved.

5.2. Future work

There are a few things that can be improved in the future

- Corpus size-translation tradeoff: Due to the high computational cost of translation, it's hard to select a corpus with extremely huge size. One method is discarding the original english corpus and choose a german corpus. The advantage is avoiding the translation problem and having a corpus with bigger size can lead to more DC-EC pairs, thus good for training. The disadvantage is that we need to build another corpus and hard to compare the performance of english and german model.

- Word embedding: Like the experiment 4.2 shows, the statified contextualized word embeddings do not have a better performance. The reason may be that fasttext deals with the word that do not occur in the training corpus better. The first method to improve the word embedding is to have a bigger training corpus. The second method is to develop an algorithm to deal with the out-of-corpus word, for example: building word embedding for small word pieces like Bert.

- Contrastive expansions: In this paper, we only focus on the consistent expansions and contrastive expansions are unfortunately not being discussed. As the paper from IBM® shows that contrastive expansions should be a more challenging task, there are probably more things to be explored.

- Filter and feature selection: There could be more and better selections for the filter and the feature. But be careful of the size of corpus, the more filters a model has the less pairs will remain for the training.

Bibliography

1. Bar-Haim, Roy, Dalia Krieger, Orith Toledo-Ronen, Lilach Edelstein, Yonatan Bilu, Alon Halfon, Yoav Katz, Amir Menczel, Ranit Aharonov, and Noam Slonim (July 2019). „From Surrogacy to Adoption; From Bitcoin to Cryptocurrency: Debate Topic Expansion". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 977–990. DOI: 10.18653/v1/P19-1094. URL: https://aclanthology.org/P19-1094.

2. Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc."

3. Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2016). „Enriching Word Vectors with Subword Information". In:

4. Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux (2013). „API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

5. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

6. Finlay, P.J. (n.d.). *Argos Translate*. Open source neural machine translation software. URL: https://www.argosopentech.com.

7. Gleim, Rüdiger, Alexander Mehler, and Sung Y. Song (2018). „WikiDragon: A Java Framework For Diachronic Content And Network Analysis Of MediaWikis". In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. accepted. Miyazaki, Japan.

8. Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov (2018). „Learning Word Vectors for 157 Languages". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

9. Guhr, Oliver, Anne-Kathrin Schumann, Frank Bahrmann, and Hans Joachim Böhme (May 2020). „Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems". In: *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 1620–1625. URL: https://www.aclweb.org/anthology/2020.lrec-1.202.

10. Gupta, Prakhar and Martin Jaggi (2021). „Obtaining Better Static Word Embeddings Using Contextual Embedding Models". In: *ACL*.

11. Hamp, Birgit and Helmut Feldweg. (1997). „GermaNet - a Lexical-Semantic Net for German." In: *Proceedings of the ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications.*

12. Heitmann, Mark, Christian Siebert, Jochen Hartmann, and Christina Schamp (2020). „More than a feeling: Benchmarks for sentiment analysis accuracy". In: *Available at SSRN 3489963*.

13. Hemati, Wahed (2020). „TextImager-VSD : large scale verb sense disambiguation and named entity recognition in the context of TextImager". PhD thesis, p. 174. URL: http://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/56089.

14. Hemati, Wahed, Tolga Uslu, and Alexander Mehler (2016). „TextImager: a Distributed UIMA-based System for NLP". In: *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science and Information Systems. Osaka, Japan.

15. Henrich, Verena and Erhard Hinrichs. (May 2010). „GernEdiT - The GermaNet Editing Tool." In: *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). , pp. 2228-2235.*

16. Honnibal, Matthew, Ines Montani, Sofie Van Landeghem, and Adriane Boyd (2020). „spaCy: Industrial-strength Natural Language Processing in Python". In: DOI: 10.5281/zenodo.1212303.

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

18. Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). *Distributed representations of words and phrases and their compositionality*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

19. Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (June 2013). „Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751. URL: https://aclanthology.org/N13-1090.

20. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). „Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

21. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). „GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

22. Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning (2020). „Stanza: A Python Natural Language Processing Toolkit for Many Human Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. URL: https://nlp.stanford.edu/pubs/qi2020stanza.pdf.

23. Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). *Language Models are Unsupervised Multitask Learners*.

24. University, Princeton (2010). *Princeton University "About WordNet."* URL: https://wordnet.princeton.edu.

25. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). „Attention is All you Need". In: *Advances in Neural Information Processing Systems*, pp. 6000–6010. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.

26. Weizenbaum, Joseph (1966). „ELIZA—a computer program for the study of natural language communication between man and machine". In: *Commun. ACM* 9, pp. 36–45.

27. Wikipedia user Tanzx30, distributed under a CC0 1.0 license (2017). *English: Hypernym/hyponym (and co-hyponym) relation*. en:File:Hyponymsandhypernyms.jpg. URL: https://en.wikipedia.org/wiki/File:Hyponymsandhypernyms.jpg.

28. Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush (Oct. 2020). „Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Appendices

A. List of Debate Topics (english)

- Gun control
- Capital punishment
- Human cloning
- Drug
- Animal testing
- Climate change
- Violence and video games
- Minumum wage
- Universal basic income
- Prostitution
- Abortion
- Bottled water
- Plastic bag
- Social media
- Artificial intelligence
- DNA
- Space colonization
- Renewable energy
- Cryptocurrency
- Fossil fuel
- Health care
- Euthanasia
- Vaccine
- Obesity
- Religion
- Homeschooling
- Democracy
- Smartphone
- Same-sex marriage
- Technology
- Summer vacation
- Globalization
- Tax
- Money
- Equal pay
- Refugee
- Patriotism
- Smoking
- Nuclear weapon
- Sport
- Art
- Vegetarian
- Capitalism
- Transgender
- Student loan

- Monopoly
- Gender equality
- Electric car
- Internet
- Doping in sport

B. List of Debate Topics (german)

- Waffenkontrolle
- Feuerwaffenverordnung
- Rüstungskontrolle
- Todesstrafe
- menschliches Klonen
- Drogen
- Medikament
- Tierversuche
- Tierforschung
- Klimawandel
- Mindestlohn
- universelles Basiseinkommen
- bedingungslose Grundeinkommen
- das Grundeinkommen der Bürger
- universelles Grundeinkommen
- Prostitution
- Abtreibung
- Flaschenwasser
- Plastiktüte
- soziale Medien
- künstliche Intelligenz
- DNA
- Deoxyribonuclesäure
- Raumkolonie
- Raumordnung
- extraterrestrische Kolonie
- erneuerbare Energie
- Kryptowährung
- fossiler Brennstoff
- Gesundheitsversorgung
- Gesundheitswesen
- Euthanasie
- Impfung
- Impfstoff
- Fettleibigkeit
- Religion
- Heimschule
- häusliche Bildung
- elektive Heimerziehung
- Demokratie
- Smartphone
- gleichgeschlechtliche Ehe
- Technologie
- Sommerurlaub
- Sommerpause
- Sommerferien
- Globalisierung

- Steuer
- Lohngerechtigkeit
- Lohngleichheit für gleiche Arbeit
- Flüchtlinge
- Patriotismus
- nationales Stolz
- Rauchen
- Atomwaffen
- Atombombe
- nukleare Bomben
- nukleare Kriege
- Sport
- Kunst
- Vegetarisch
- Kapitalismus
- Transgender Menschen
- Studentendarlehen
- Monopol
- Geschlechtergleichstellung
- sexuelle Gleichheit
- Gleichstellung der Geschlechter
- Elektroauto
- Batterie-Elektrofahrzeug
- Internet
- Doping

C. List of Extraction Patterns

The list of extraction patterns are as follows:

- DC ist ein EC
- DC ist eine EC
- DC ist eine Art [des|der|von|vom] EC
- DC ist eine Form [des|der|von|vom] EC
- DC ist ein Spezialfall [des|der|von|vom] EC
- DC oder andere Arten [des|der|von|vom] EC
- DC oder eine andere Art [des|der|von|vom] EC
- DC oder [ander|andere|anderes] EC
- DC und [ander|andere|anderes] EC
- DC und andere Arten [des|der|von|vom] EC
- DC ist ein Beispiel für EC