# NLP project - final report
# SLR - NieLeniweProjekty

**First Author: Michał Gozdera**
Warsaw University of Technology
01142172@pw.edu.pl

**Second Author: Małgorzata Hadasz**
Warsaw University of Technology
01156169@pw.edu.pl

**Third Author: Krystian Kurek**
Warsaw University of Technology
01121582@pw.edu.pl

**supervisor: Anna Wróblewska**
Warsaw University of Technology
anna.wroblewska1@pw.edu.pl

## Abstract

Nowadays, the rapid increase in knowledge and the amount of performed research in numerous domains (like Computer Science and Medicine) causes the need for solutions designed to segregate, organize and find created papers and publications automatically. A key aspect of such frameworks is to detect the topic of a given article and connect the discovered subject with domain-specific concepts. Hence, the aim of our project was to address the question of finding semantic keywords for a systematic literature overview. Namely, we propose different solutions to extract keywords from medical papers abstracts, tag these keywords with ontologies concepts and choose the best tags based on disambiguation techniques.

For keywords extraction, we focus mainly on BERTopic (Grootendorst, 2022b) model, but other we also compare it with LDA (Blei et al., 2003) method.

One approach to keyword tagging will be performed with the use of NBCO annotator (Jonquet et al., 2009) (standard and simple solution). At the same time, the other will be implemented from scratch with the use of the word embedding concept (*word2vec* (Mikolov et al., 2013)).

Tag disambiguation techniques rely on the Closest Sense method (Alexopoulou et al., 2009), adjusted to our problem statement.

To the best of our knowledge, there is currently no state-of-the-art solution combining the three functionalities mentioned above and taking advantage of the latest NLP solutions. Authors of (van Dinter et al., 2021) mention the 12 steps that are defined in Systematic Literature Overview (SLR) domain They are divided in to 3 main categories, which are: Need for a review, Conducting the review, Reporting the review. The claim to find 41 studies approaching to automate one or more selected steps in the SLR process, mainly for the Software Engineering and Medical domain. Our project focus on Conducting the review. We try to automate steps: Identification of research, Selection of primary studies, Study quality assessment by keywords extraction and ontology tagging. What differs our approach from the ones currently available is the use of BERTopic to extract keywords from papers. What is more, the combination of keywords extraction, ontology tagging and disambiguation was not covered in the literature to the best of our knowledge.

| Work assigment | Person |
|---|---|
| BERTopic | M. Gozdera (11.11%) |
| NCBO tagger | M. Gozdera (11.11%) |
| Disambiguation research | M. Gozdera (11.11%) |
| UMLS extraction | K. Kurek (11.11%) |
| Embedding tagger | K. Kurek (11.11%) |
| Statistics summary | K. Kurek (11.11%) |
| LDA | M. Hadasz (11.11%) |
| Data analysis | M. Hadasz (11.11%) |
| Disambiguation implementation | M. Hadasz (11.11%) |

Table 1: Table of contents

## 1 Introduction

Medicine's scientific area is characterized by a rapid pace of creating new literature. Every year, numerous new papers in different medical domains are produced. On the other hand, reliable research requires authors to be familiar with current achievements, which causes the need for effective and exact literature searching.

The scientific goal of our research was to create a solution that would help researchers and people interested in medical papers to find the most suitable scientific publications according to their needs.

The research question we want to address is whether the solution described above can be prepared using the latest achievements in the NLP domain. Our proposition is based on several techniques, including keywords extractors, ontology-based taggers and disambiguation algorithms.

In the rest of this section, we describe the significance of the project, project activities, timeline, specific research goals, and post-hoc risk analysis. Section 2 describes related works (state-of-the-art and data used). Section 3 is dedicated to descriptions of the methods we used. Section 4 describes the methods of evaluation we incorporated. In Section 5, we analyze the results and discuss the challenging parts of the project. We conclude in section 6, also providing the possibilities for future work. In Appendix A, we describe in detail the methodology of the techniques used.

## 1.1 Significance of the project

To the best of our knowledge, there is no obvious solution that would define an end-to-end process of semantic keywords for systematic literature reviews. Based on our research, there exist models and algorithms that can be successfully modified and combined to create a well-performing method tackling the scientific problem we describe.

Serveral approaches to Semantic Literature Reviews are present in the literature (Bashir and Warraich, 2020)(van Dinter et al., 2021). The nature of our project includes not only combing existing solutions into one designed for a specific problem but also a modification of existing methods (Closest-Sense method modification described in the appendix) and implementing our own ideas from scratch.

We contribute to the research field in two areas:

- research resulting in creating a method of extracting semantic keywords for systematic literature reviews, based on recent NLP achievements, preparing a framework to test the method based on publicly available datasets (CRAFT, MedMentions) with the use of metrics described in 4,
- developing an actual solution and implementation that can be successfully incorporated into the research community and improve the quality and speed of research work. Extracting keywords and tagging them is a factor that can help in finding scientific papers faster. The use of tags from ontologies makes the process unified and more accurate.

| Date | Stage name | Description |
|---|---|---|
| 4.11.2022 | Project proposal | literature review, solution concept and proposal |
| 18.11.2022 | Proof of concept | exploratory data analysis and preliminary machine learning models |
| 9.12.2022 | Final project | full solution and prepared product |

Table 2: Project activity and timeline

## 1.2 Project activities and timeline

We divided our project into 3 main parts, presented in Table 2.

## 1.3 Specific research goals

We established the following general research goals for the project:

- acquiring wide knowledge about current state-of-the-art methods in NLP domain, especially in semantic keywords for systemic literature reviews field,
- testing different solutions currently available and adapting them to the above need,
- combining existing methods for keywords retrieval, keywords ontology annotating and tags disambiguation into one working solution,
- creating new algorithms for keywords ontology annotating and tags disambiguation.

More specifically, we divided our work into the following parts:

- investigation of available solutions of keywords extractors (LDA, BERTopic). Tests and comparisons between solutions. The performance and efficiency analysis.

- examining an available tagging solution - NCBO. Determining the way to communicate with NCBO rest API and preparing solutions to connect with it from Python code. Preparing Python API to automatically annotate keywords with NCBO,

- preparing word embedding tagger by implementing it from scratch (in Python) based on BioBert (Lee et al., 2019) embeddings,

- implementing Closest Sense method for applying disambiguation technique (from scratch in Python).

### 1.4 Risk analysis

After working on and developing the project, we want to summarize the risks we predicted and experienced. Concerning the first two identified risks in privacy policies of algorithms or datasets, we did not experience any of those threads. For both data and algorithms, the privacy policies did not change, and therefore we were able to use them. Also, with regard to the risk to the team, we did not experience any problems with teamwork. All the members worked equally and with engagement and did not make any serious mistakes. The risk that we expected and experienced is connected to a time shortage. Even though we managed to finish the project on time, we think that with proper hyperparameter tunning, initial topics sorting and experimenting with the subset of keywords, we can increase the performance results. Therefore, as mentioned in the analysis, we plan to implement those changes in the next assignment.

## 2 Related works

### 2.1 Current solutions and state-of-the-art

Currently, available solutions are not specifically directed toward the aim we presented. There exist state-of-the-art solutions performing specific parts of what we are going to implement, but the entire process itself is not well investigated, in our opinion.

Regarding keywords extraction, Latent Dirichlet Allocation - LDA (Blei et al., 2003) is one of the state-of-the-art algorithms. However, it is usually replaced by models utilizing modern word embedding techniques, like BERTopic (Grootendorst, 2022b).

For a long time, simple solutions for ontology-based tagging in medical data, like NCBO annotator (Jonquet et al., 2009) were used. Recently, more sophisticated approaches (like ScispaCy (Neumann et al., 2019)) appeared. What is more, the word embedding idea is getting more and more interest in the NLP field, actually being the current state-of-the-art word representation (mainly because of its high performance and important properties, like preserving semantic meaning). However, to the best of our knowledge, no state-of-the-art embedding-based annotator is provided for the use of specific medical ontologies.

According to our research, ontology tag disambiguation is the least explored part of the solution. There are not many papers approaching this topic, most of them treating disambiguation as a side part of other solutions ((Leaman and Lu, 2016), (Bindelli et al., 2008a)). An algorithm that seems to fit the needs of our solution best is the Closest Sense method (Alexopoulou et al., 2009).

### 2.2 Results of preliminary research

Introductory research resulted in gathering knowledge about state-of-the-art methods than can be incorporated into our solution. They are described in section 2. Apart from reading about particular solutions, we also tested and verified existing implementations:

- Keywords extraction - the LDA algorithm is available in *sklearn* library (Buitinck et al., 2013); BERTopic can be found in *bertopic* package (Grootendorst, 2022a),

- Keywords tagging – for NCBO annotator, the REST API is available (Jonquet et al., 2021), so our solution is based on the use of it via HTTP connection; there is no concrete implementation of embedding-based annotations with medical ontologies that would satisfy our need, so this part is implemented from scratch based on *word2vec* implementation (*gensim* package (Rehurek, 2022)),

- Tags disambiguation – since we are going to use a modified version of the Closest Sense method, we implemented it from scratch.

The main result of the preliminary research was that currently available solutions (with quite a few modifications) should allow us to develop a fully usable method for extracting semantic keywords. However, no specific algorithms pipeline (combining all the above methods) is available now. Creating one was the goal of our project.

### 2.3 Data

To develop and test our solution, we used the MedMentions data set (Mohan and Li, 2019). This resource provides access to over 4,000 articles (titles and abstracts) published in PubMed. Each article is annotated with UMLS (Bodenreider, 2004) concepts by professional annotators with rich experience in biomedical content. We treated those articles and annotations as the Gold Standard. We decided to use a subset of MedMentions dataset - MedMentions ST21pv. It is annotated with only 18 ontologies (chosen by the authors of MedMentions by their relevance) and uses only 21 Semantic Types (out of 127 available in MedMentions). In UMLS, semantic types are more general than concepts. Each concept can be assigned to several semantic types, and each semantic type can have (and actually always has) multiple concepts assigned.

In part of our solution, we needed to provide ontologies. We focused on UMLS ontology (it is actually a *super-ontology* cause it gathers together various ontologies from the medical domain). As mentioned above, we trained and evaluated our solution on MedMentions ST21pv, meaning we took 18 ontologies chosen by MedMentions authors.

### 2.4 Exploratory data analysis

As a dataset, we decided to use **MedMentions**(Mohan and Li, 2019). It contains 2 datasets:

- Full – containing all annotations
- ST21pv – with a subset of the full annotations, targeting information retrieval

The **MedMentions**(Mohan and Li, 2019) contains 4392 abstracts and titles selected from PubMed, released between January 2016 and January 2017 in English and regarding the biomedical field. Each document was annotated by experienced, professional annotators, who marked all UMLS(Bodenreider, 2004) concepts mentioned in these documents. Each concept is linked to at least one Semantic Type. Authors of the dataset claim that UMLS (Bodenreider, 2004) contains many concepts that are not useful for specialized document retrieval. Therefore, after creating the full dataset, they prepared the ST21pv one. In this version:

1. All concepts linked to semantic types 1 or 2 are eliminated (too broad).
2. Based on biomedical relevance, they selected 21 semantic types at levels 3-5 – only concepts mapped to them (or their descendants) were considered.
3. They selected 18 preferred source vocabularies and excluded all concepts that were not linked to those sources.

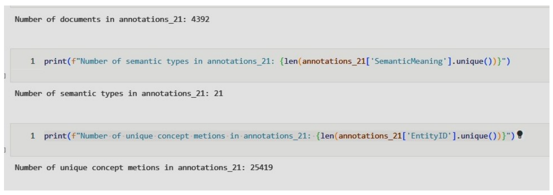
As the authors suggested, we decided to use the ST21pv dataset. The dataset contains two types of information: About data:

1. Id of the document,
2. Type (a for abstract, t for a title),
3. Content - text of the abstract/title.

About annotations:

1. Id of the document,
2. StartIndex, EndIndex – indices into the document text,
3. MentionTextSegment – the actual mention of the concept,
4. EntityId – the UMLS (Bodenreider, 2004) entity id,
5. SemanticType ID – the id of the semantic type the entity is linked to.

Firstly, we checked if all numbers stated in the paper (Mohan and Li, 2019) are correct. According to the authors, there are 25.419 unique concepts mentioned, 203.282 concepts mentioned, and 4392 documents; all those numbers are consistent with the one given in paper (Mohan and Li, 2019). We also verified if both title and abstract exist for every document ID, and it agreed (Figure 1).



```
Number of documents in annotations_21: 4392

print(f"Number of semantic types in annotations_21: {len(annotations_21['SemanticMeaning'].unique())}")

Number of semantic types in annotations_21: 21

print(f"Number of unique concept metions in annotations_21: {len(annotations_21['EntityID'].unique())}")

Number of unique concept metions in annotations_21: 25419
```

Figure 1: Given numbers verification

### 2.5 Data preprocessing

We decided to preprocess data in a format which will be useful for the algorithms. Firstly, we divided each text into sentences (using the *sent_tokenize* function from the nltk library (Bird et al., 2009)). We tokenized each sentence into words (using the *word_tokenize* function from the nltk library (Bird et al., 2009)) and removed the stop words. Subsequently, we perform stemming (using *PorterStemmer* from nltk library (Bird et al., 2009)) and lemmatization (using *WordNetLemmatizer* from nltk library (Bird et al., 2009)). As a result, we got 5 new columns: *tokenized_sentences*, *tokenized_words*, *tokenized_words_no_stopwords*, *tokenized_words_processed* and *tokenized_words_lemmatize*. Ultimately, we divided the dataset into train and test sets (8 to 2 ratio). Moreover, we mapped semantics to their meanings, which is useful for interpretation. Additionally, we prepared two new tables: one containing the list of all semantic types mentioned in each document, and the other containing all concepts mentioned in each document.

## 3 Approach & research methodology

Our approach and research methodology consist of several steps.

Firstly, we aimed to perform thorough research in both general NLP and methods specific to our problem.

The second part of the problem investigation was to test various implementations of currently available methods.

Next, we prepared a Proof of Concept (PoC) solution that utilized some of the concepts included in previous sections. It will probably be composed of ready-to-use solutions, like BERTopic + NCBO annotator. It aimed to illustrate the way system will work.

Then, we prepared the final solution, including all methods described in previous sections. Despite the fact that we reserved some space for potential changes during the project development, we used all the methods we planned and implemented the solution according to initial ideas.

Each stage of the project was presented in front of other researchers working on similar projects and the project supervisor.

## 4 Methods of results analysis

To compare tagging methods, we will use two metrics: precision, recall, and F1 score, which are defined as follows:

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

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

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{3}$$

Where:

- True positives (TP) - number of cases when our annotation matches annotation from the golden standard dataset
- False positives (FP) - number of cases when our annotation does not match annotation from the golden standard dataset
- False negatives (FN) - number of cases when annotation from the golden standard dataset is not present in our annotations

It is hard to come up with a way to calculate the number of true negatives. That is why we choose metrics that do not rely on this particular number.

## 5 Solution analysis, result discussion and tests

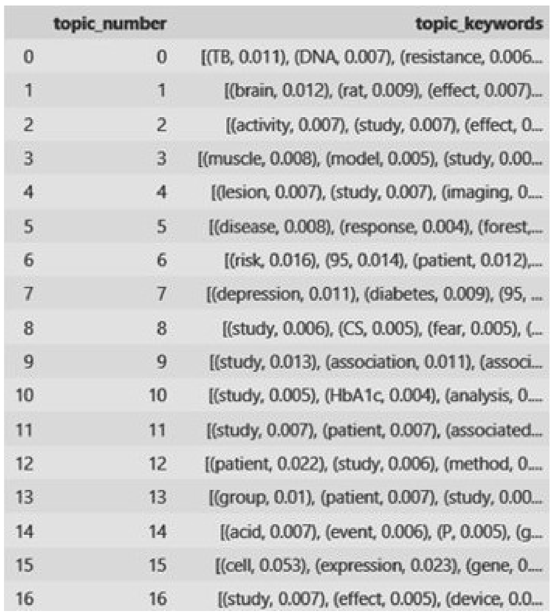
This section includes a description of the final implementation, tests, and obtained results.

### 5.1 Keyword extraction

#### 5.1.1 LDA

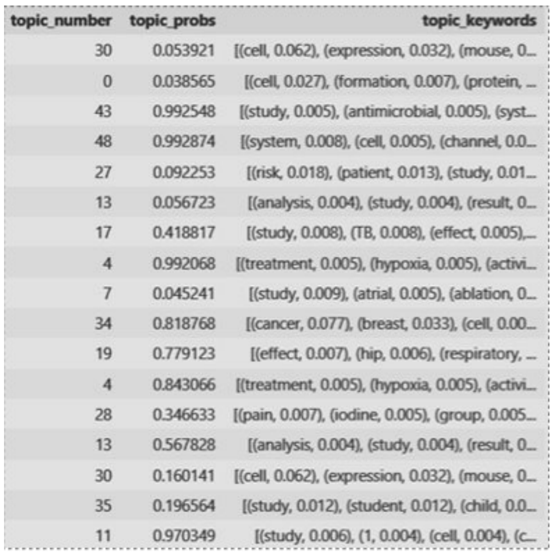
We used the algorithm implemented in the *genism* (Rehurek and Sojka, 2011) package. As a first trial, we decided to train the LDA on 59 topics (this is the number of topics retrieved by *BERTopic* A.1.2); as input data, we used tokenized and lemmatized ( with stop words removal) data.

For each topic, we got the list of words that described it, with the probability of its occurrence, and for each document, we got the list of topics, with their probabilities. In this project, we assigned the best ( the most probable) topic to each document. As a result of this part of the project, we produced the file, in which for each document, we assigned the topic, the probability of this topic and the list of its words. The results obtained by the LDA algorithm are presented in Figure 2 and Figure 3. We think it would be worth investigating the different number of predefined topics to improve LDA's performance. Therefore, we decided to make it one of our goals for future development.



| | topic_number | topic_keywords |
|---|---|---|
| 0 | 0 | [(TB, 0.011), (DNA, 0.007), (resistance, 0.006… |
| 1 | 1 | [(brain, 0.012), (rat, 0.009), (effect, 0.007)… |
| 2 | 2 | [(activity, 0.007), (study, 0.007), (effect, 0… |
| 3 | 3 | [(muscle, 0.008), (model, 0.005), (study, 0.00… |
| 4 | 4 | [(lesion, 0.007), (study, 0.007), (imaging, 0… |
| 5 | 5 | [(disease, 0.008), (response, 0.004), (forest,… |
| 6 | 6 | [(risk, 0.016), (95, 0.014), (patient, 0.012),… |
| 7 | 7 | [(depression, 0.011), (diabetes, 0.009), (95, … |
| 8 | 8 | [(study, 0.006), (CS, 0.005), (fear, 0.005), (… |
| 9 | 9 | [(study, 0.013), (association, 0.011), (associ… |
| 10 | 10 | [(study, 0.005), (HbA1c, 0.004), (analysis, 0… |
| 11 | 11 | [(study, 0.007), (patient, 0.007), (associated… |
| 12 | 12 | [(patient, 0.022), (study, 0.006), (method, 0… |
| 13 | 13 | [(group, 0.01), (patient, 0.007), (study, 0.00… |
| 14 | 14 | [(acid, 0.007), (event, 0.006), (P, 0.005), (g… |
| 15 | 15 | [(cell, 0.053), (expression, 0.023), (gene, 0… |
| 16 | 16 | [(study, 0.007), (effect, 0.005), (device, 0.0… |

Figure 2: LDA: Topics and their keywords for corpus based on MedMentiones (Mohan and Li, 2019)



| topic_number | topic_probs | topic_keywords |
|---|---|---|
| 30 | 0.053921 | [(cell, 0.062), (expression, 0.032), (mouse, 0… |
| 0 | 0.038565 | [(cell, 0.027), (formation, 0.007), (protein, … |
| 43 | 0.992548 | [(study, 0.005), (antimicrobial, 0.005), (syst… |
| 48 | 0.992874 | [(system, 0.008), (cell, 0.005), (channel, 0.0… |
| 27 | 0.092253 | [(risk, 0.018), (patient, 0.013), (study, 0.01… |
| 13 | 0.056723 | [(analysis, 0.004), (study, 0.004), (result, 0… |
| 17 | 0.418817 | [(study, 0.008), (TB, 0.008), (effect, 0.005),… |
| 4 | 0.992068 | [(treatment, 0.005), (hypoxia, 0.005), (activi… |
| 7 | 0.045241 | [(study, 0.009), (atrial, 0.005), (ablation, 0… |
| 34 | 0.818768 | [(cancer, 0.077), (breast, 0.033), (cell, 0.00… |
| 19 | 0.779123 | [(effect, 0.007), (hip, 0.006), (respiratory, … |
| 4 | 0.843066 | [(treatment, 0.005), (hypoxia, 0.005), (activi… |
| 28 | 0.346633 | [(pain, 0.007), (iodine, 0.005), (group, 0.005… |
| 13 | 0.567828 | [(analysis, 0.004), (study, 0.004), (result, 0… |
| 30 | 0.160141 | [(cell, 0.062), (expression, 0.032), (mouse, 0… |
| 35 | 0.196564 | [(study, 0.012), (student, 0.012), (child, 0.0… |
| 11 | 0.970349 | [(study, 0.006), (1, 0.004), (cell, 0.004), (c… |

Figure 3: LDA: Documents from MedMentiones (Mohan and Li, 2019) with the number of the best topic and its probability

### 5.1.2 BERTopic

We used the implementation from the dedicated *bertopic* package. We used the basic SentenceBert model as the words embedding model present in BERTopic. We tested different approaches to data preparation before applying the BERTopic model:

- raw data (no preprocessing),
- stop words removal,
- stop words removal and stemming,
- stop words removal and lemmatization.

Theoretically, BERTopic should also work fine with unprocessed text, as it is capable of detecting outliers (in the HDBSCAN clustering algorithm we used). Our experiments showed that, indeed, BERTopic applied to raw data created an *outlier topic* that was represented by stop words (*a, the, an, and, or* . . . ), but stop words were also present in other topics representations, which resulted in assuming they are the keywords messing up the final solution.

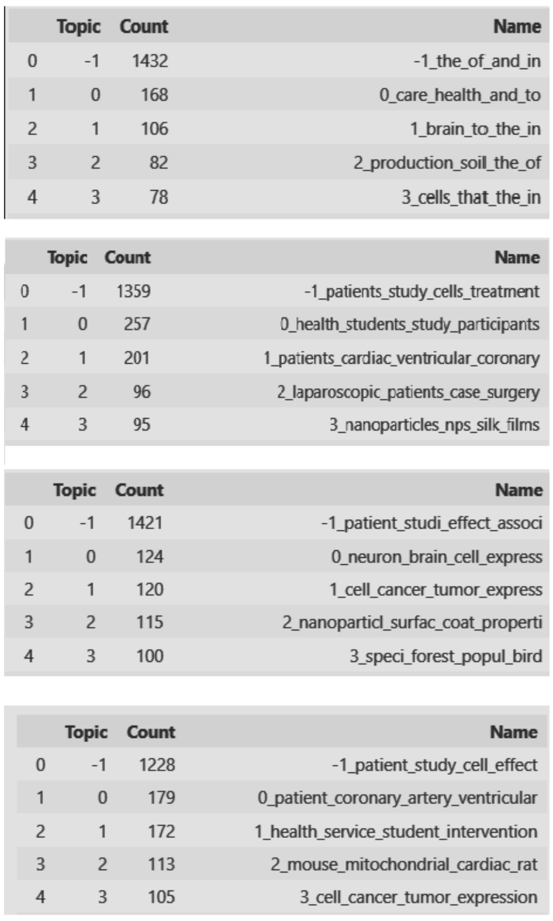
Applying stemming also did not provide good results since the extracted keywords were not full words (correct in English). That is why the next step (ontology tagging) was affected. Taggers could not match stems with ontologies concept very well.

In the case of applying only stop words removal, sometimes happened that found keywords represented actually the same words but in different forms (like *image, images*).

We concluded that the best option was to apply only stop word removal and lemmatization since it handled all problems mentioned above (stop words as keywords, different forms of the same words). It also produced the lowest number of outliers.

Figure 4 shows exemplary topics for each approach described above. We can notice *to, in, of* as keywords for actual topics for raw data. For stemmed data, we see keywords being incomplete words (stems only), like *surfac*.

An important fact is that quite a lot of training papers were clustered as outliers (-1 topic): 1228-1432 out of 3513. The number of outliers was similar for different values of BERTopic hyperparameters tested and is considered a limitation of BERTopic. In the next project, we also aim to test a different number of words needed to form a topic and verify whether it influences the number of outliers.



| | Topic | Count | Name |
|---|---|---|---|
| 0 | -1 | 1432 | -1_the_of_and_in |
| 1 | 0 | 168 | 0_care_health_and_to |
| 2 | 1 | 106 | 1_brain_to_the_in |
| 3 | 2 | 82 | 2_production_soil_the_of |
| 4 | 3 | 78 | 3_cells_that_the_in |

| | Topic | Count | Name |
|---|---|---|---|
| 0 | -1 | 1359 | -1_patients_study_cells_treatment |
| 1 | 0 | 257 | 0_health_students_study_participants |
| 2 | 1 | 201 | 1_patients_cardiac_ventricular_coronary |
| 3 | 2 | 96 | 2_laparoscopic_patients_case_surgery |
| 4 | 3 | 95 | 3_nanoparticles_nps_silk_films |

| | Topic | Count | Name |
|---|---|---|---|
| 0 | -1 | 1421 | -1_patient_studi_effect_associ |
| 1 | 0 | 124 | 0_neuron_brain_cell_express |
| 2 | 1 | 120 | 1_cell_cancer_tumor_express |
| 3 | 2 | 115 | 2_nanoparticl_surfac_coat_properti |
| 4 | 3 | 100 | 3_speci_forest_popul_bird |

| | Topic | Count | Name |
|---|---|---|---|
| 0 | -1 | 1228 | -1_patient_study_cell_effect |
| 1 | 0 | 179 | 0_patient_coronary_artery_ventricular |
| 2 | 1 | 172 | 1_health_service_student_intervention |
| 3 | 2 | 113 | 2_mouse_mitochondrial_cardiac_rat |
| 4 | 3 | 105 | 3_cell_cancer_tumor_expression |

Figure 4: Five first topics extracted with BERTopic for raw data (first table), data after stop word removal (second table), data after stemming and stop words removal (third table), data after lemmatization and stop words removal. $-1$ topic means the *outlier topic*, *Count* is the number of documents from the training set that were assigned to a given topic, *Name* shows top 4 keywords representing the topic.

We ran BERTopic that aimed to find the top 10 keywords for each paper and set the minimum number of words in a single cluster to 10. We used titles concatenated with abstracts (without stop words) as input data.

The time of keywords extraction for all train and test examples (around 4000 papers) was quite short (15-19 minutes, on CPU Intel Core i7, 10th gen). We checked the results on several exemplary abstracts and concluded the outcome keywords are relevant and suitable. BERTopic turned out to be quite a powerful and efficient solution for keyword extraction.

We tested a few sets of values of BERTopic hyperparameters but did not notice any significant difference in produced outputs. In the production environment (with suitable computing resources and time), BERTopic model could be fine-tuned within the entire algorithm pipeline (keywords extraction, tagging, disambiguation).

We extracted 59 different topics from the training set, each with 10 keywords assigned.

## 5.2 Words tagging

### 5.2.1 NCBO tagger

For the purpose of this project, we use NCBO REST API to tag keywords extracted from papers. REST communication is an important limitation since annotating all samples requires a lot of time (training set - 3,513 examples - around 80 minutes).

Since, in the project, we used the ST21pv dataset, we needed to restrict the NCBO ontologies to those used for annotating ST21pv. The same applies to semantic types - we should use only concepts of 21 semantic types used for creating ST21pv. However, this step caused NCBO to find a very small number of annotations. We compare the results obtained (counts of annotations for a given paper) in Figure 5. The results are based on tagging keywords extracted from BERTopic trained on data with stop words removed and lemmatization applied.

We performed analogous tests for stemmed data without stop words, but in this case, the number of found tags was, unsurprisingly, even smaller (Figure 6).

During tests of NCBO, we found a few inconsistencies in the NCBO annotator API and MedMentions data. NCBO API often returned too few tags when limiting the semantic types space to 21 defined for ST21pv MedMentions. That is why in further steps of this project, we decided to use all semantic types for creating NCBO tags (as can be seen in Figures 5 and 6 using only 21 semantic types would not allow for any relevant tagging).

We extract each keyword's potential direct (concepts from ontologies) and more general (concept ancestors) tags.

As mentioned before, NCBO is not time-

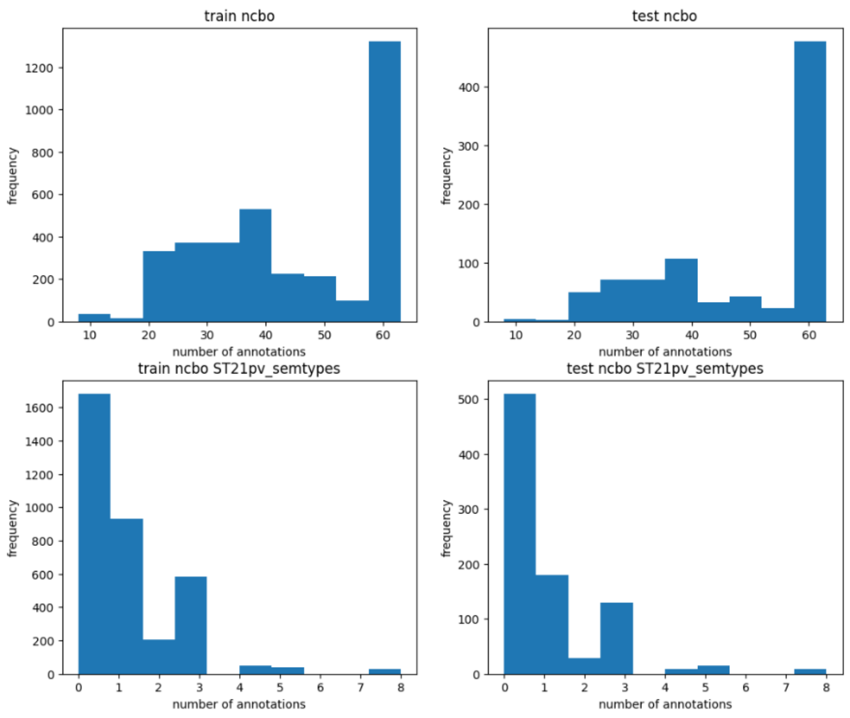


Figure 5: Histograms of a number of annotations found by NCBO tagger for all semantic types (*train ncbo*, *test ncbo*) and limited to semantic types used in ST21pv (*train ncbo ST21pv_semtypes*, *test ncbo ST21pv_semtypes*). Input data were keywords extracted by BERTopic on data with stop words removed and **lemmatization** applied.

efficient since it needs REST API communication. However, considering that we needed to only tag keywords (not the entire abstracts), its time performance was acceptable. (around 80 minutes to tag the training set and around 30 minutes to tag the train set).

#### 5.2.2 Embedding tagger

We needed help with the absence of approximately 65% of words from abstracts in word2vec pre-trained models. We decided to switch to the BERT model that provides out-of-vocabulary solutions. We have used pre-trained bioBERT from the transformers library. We calculated the embeddings of keywords extracted by BERTopic and embeddings of concept names from UMLS ontology. We paired them against each other and assigned a concept to every keyword, choosing a concept with maximum cosine similarity. Examples of matched pairs are shown in Figure 3.

### 5.3 Disambiguation

As described in A.3 to perform word disambiguation, we used the *Keywords-based Closest Sense method*. It is a modification of the *Sentence-based Closest Sense method*, in which we compare the distance between the keyword's concepts.

| | Keyword | Concept name |
|---|---|---|
| trivial | depression | Depression |
| wrong | mitochondrial | Palestinian |
| useful | imaging | analysis |

Table 3: Table showing examples of matched concepts' names with embeddings.

To obtain those distances, we calculate embeddings of each concept using *BioBERT model*. We initially did not sort the dictionary and for each concept, assign the distance equal to 0. Then we performed the iterative process (as described in A.3). We stopped the loop when the rank of the words stopped changing since the further iteration would not bring any differences. The number of iterations needed to compute the final dictionary is presented in Figure 7. To visualize the changes in the order of the word we investigate how the changes occurred for the keyword IMPLANT. The results are presented on the Figure 8. As a result, from this part of the project, we produced the tables, where for each document, we produce the dictionary of keywords with all their possible con-

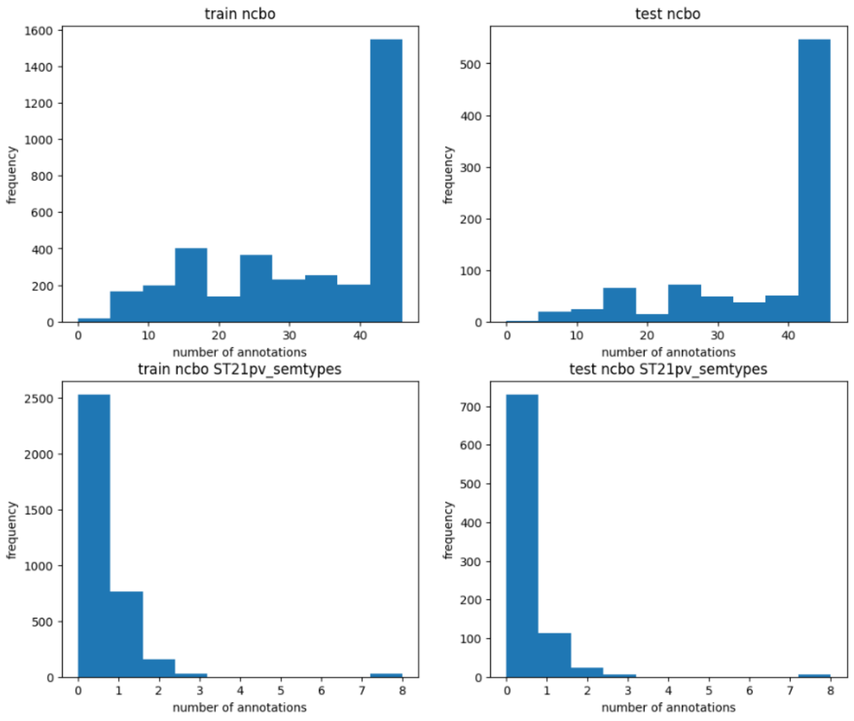


Figure 6: Histograms of number of annotations found by NCBO tagger for all semantic types (*train ncbo*, *test ncbo*) and limited to semantic types used in ST21pv (*train ncbo ST21pv_semtypes*, *test ncbo ST21pv_semtypes*). Input data were keywords extracted by BERTopic on **stemmed** data with stop words removed.

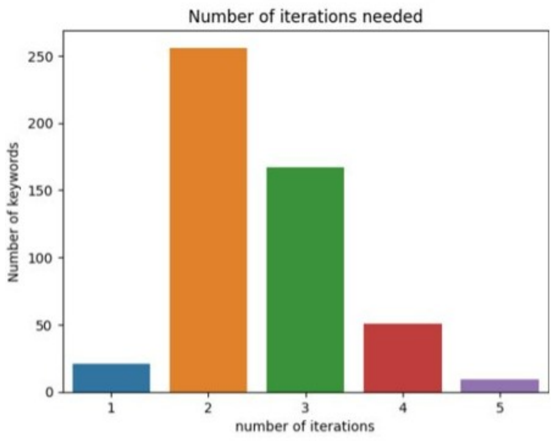


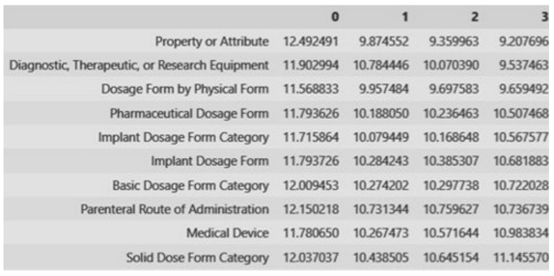
Figure 7: Number of iterations need to calculate the final result

cepts and the dictionary keyword: the best concept.

## 5.4 Difficulties and ways to overcome them

We experienced many challenging tasks during this project, not trivial to overcome.

Firstly, the scope of the project was vast. It consisted of three separate tasks (keywords extraction, tagging, and disambiguation). Each of these parts



| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Property or Attribute | 12.492491 | 9.874552 | 9.359963 | 9.207696 |
| Diagnostic, Therapeutic, or Research Equipment | 11.902994 | 10.784446 | 10.070390 | 9.537463 |
| Dosage Form by Physical Form | 11.568833 | 9.957484 | 9.697583 | 9.659492 |
| Pharmaceutical Dosage Form | 11.793626 | 10.188050 | 10.236463 | 10.507468 |
| Implant Dosage Form Category | 11.715864 | 10.079449 | 10.168648 | 10.567577 |
| Implant Dosage Form | 11.793726 | 10.284243 | 10.385307 | 10.681883 |
| Basic Dosage Form Category | 12.009453 | 10.274202 | 10.297738 | 10.722028 |
| Parenteral Route of Administration | 12.150218 | 10.731344 | 10.759627 | 10.736739 |
| Medical Device | 11.780650 | 10.267473 | 10.571644 | 10.983834 |
| Solid Dose Form Category | 12.037037 | 10.438505 | 10.645154 | 11.145570 |

Figure 8: Changes for keyword IMPLANT

was time-consuming in the case of the knowledge needed to acquire and the test we needed to perform. For example, NCBO communication via the REST API took quite a lot of time and needed an Internet connection. Also, the data analysis and ontologies investigation took more time than expected.

Another issue we needed to cope with was the inconsistencies between the UMLS ontologies set and the MedMentions dataset. When we filtered concepts from UMLS, we got concept names only for 25% of concepts used as ground truth in MedMentions. In can be caused by the fact that they use an archived historical version of UMLS concepts (MedMentions uses the version released in

2017). Even though we downloaded and analyzed version 2017AA, we cannot be 100% sure that we have extracted the correct concepts. That caused the matching concept task challenging, as we needed to put much effort into examining concepts IDS (CUIs).

The way concepts are defined in UMLS is also relatively complex. Many concept names for one concept – a problem of choosing the best one. To resolve this problem, we have decided to use a concept name most similar to all other concept names within the scope of a single concept. The measure of similarity was simple SequenceMatcher from the built-in Python library - difflib. We could not use more sophisticated methods like BERT encodings because there were too many concepts' names to resolve (almost ten million). The best way to explain how SequenceMatcher works is to show an example. Let's say we have two strings:

- s1 = "This is a string."
- s2 = "This is not a string."

First, we want to divide those strings into matching and non-matching blocks.

1. "This is" (matching)
2. " not" (non-matching)
3. " a string." (matching)

Then we count the number of characters in matching blocks, 17. We multiply it by 2 because those characters are both in **s1** and **s2**, and we get 34. In the end we get similarity measure by following calculation $similarity = \frac{34}{len(s1)+len(s2)} = 0.87$.

It was also difficult to evaluate the performance of the NBCO tagger because we used concept ID from UMLS ontologies when comparing ground truth to tagged concepts. Unfortunately, NCBO rarely provides those IDs, and we had to match them by concepts' names, and we have already discussed difficulties with them.

Finally, the task we established as a goal of this project is, by definition, very complex. Unlike a regular classification or regression task, assigning tags to keywords (or abstracts) is more subjective; hence, it isn't easy to fit into the ground truth definitions. Also, the fact that some tags did not fit into the ground truth defined by a specialist does not mean that they are incorrect assignments (they can be, for example, more general or more specific but still correct). That is why a demanding part of the project was the evaluation, and the fact that we needed to evaluate all three steps together made it even more difficult.

### 5.5 Final results

In table 4, we provided the evaluation of our methods - three metrics: recall, precision, and F1 score. Results obtained using NCBO tagger are low due to the difficulties described earlier. We achieved the best results for LDA keyword extraction with embedding tagger - 46%.

## 6 Conclusions and future work

Despite the task's immense scope and considerable complexity, we prepared a working solution, comparing different models and ways to combine them. We do think that we achieved our goals and created a solid basis for the future development of similar solutions.

For future work, we propose to introduce a few improvements to the existing solution, namely

- disambiguation improvement (initial sorting of tags according to probabilities returned by tagging tools),
- higher number of topics, different numbers of keywords for a given topic (e.g., to reduce the number of outliers)
- considering keywords not only from the best topic (to increase the number of keywords produced).

Moreover, we plan to add a mechanism of paper searching based on user queries and extracted keywords.

## References

Dimitra Alexopoulou, Bill Andreopoulos, Heiko Dietze, Andreas Doms, Fabien Gandon, Jörg Hakenberg, Khaled Khelif, Michael Schroeder, and Thomas Wächter. 2009. Biomedical word sense disambiguation with ontologies and metadata: automation meets accuracy. *BMC Bioinformatics*, 10(1):28, Jan.

Faiza Bashir and Nosheen Fatima Warraich. 2020. Systematic literature review of semantic web for distance learning. *Interactive Learning Environments*, 0(0):1–17.

| Keyword extraction | Tagger | Dataset | Precision | Recall | F1 |
|---|---|---|---|---|---|
| BERTopic | Embedding | test | 35.97 | 39.02 | 37.43 |
| | Embedding | train | 36.17 | 37.76 | 36.95 |
| | NCBO | test | 0.46 | 57.47 | 0.92 |
| | NCBO | train | 0.46 | 49.78 | 0.91 |
| LDA | Embedding | test | 47.90 | 44.37 | **46.07** |
| | Embedding | train | 42.37 | 40.22 | 41.27 |
| | NCBO | test | 2.41 | 39.92 | 4.55 |
| | NCBO | train | 3.18 | 29.65 | 5.74 |

Table 4: Table showing results of different keyword extraction and tagging methods.

Silvia Bindelli, Claudio Criscione, Carlo Curino, Mauro Drago, Davide Eynard, and Giorgio Orsi. 2008a. Improving search and navigation by combining ontologies and social tags. pages 76–85, 11.

Silvia Bindelli, Claudio Criscione, Carlo Curino, Mauro Drago, Davide Eynard, and Giorgio Orsi. 2008b. Improving search and navigation by combining ontologies and social tags. pages 76–85, 11.

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

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl$_1$) : $D267 - -D270, 01$.

Lars Buitinck, 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, URL: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html`.*

Maarten Grootendorst. 2022a. Bertopic - pypi, url: `https://pypi.org/project/bertopic/`.

Maarten Grootendorst. 2022b. Bertopic: Neural topic modeling with a class-based tf-idf procedure.

Clement Jonquet, Nigam Shah, Cherie Youn, Mark Musen, Chris Callendar, and Margaret-Anne Storey. 2009. Ncbo annotator: Semantic annotation of biomedical data. *ISWC*, 01.

Clement Jonquet, Nigam Shah, Cherie Youn, Mark Musen, Chris Callendar, and Margaret-Anne Storey. 2021. Ncbo annotator rest api, url: `https://bioportal.bioontology.org/annotator`, 01.

İlknur Karadeniz and Arzucan Özgür. 2019. Linking entities through an ontology using word embeddings and syntactic re-ranking. *BMC Bioinformatics*, 20(1):156, Mar.

Robert Leaman and Zhiyong Lu. 2016. TaggerOne: joint named entity recognition and normalization with semi-Markov Models. *Bioinformatics*, 32(18):2839–2846, 06.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Sunil Mohan and Donghui Li. 2019. Medmentions: A large biomedical corpus annotated with umls concepts.

Claire Nédellec, Robert Bossy, Estelle Chaix, and Louise Deléger. 2018. *Text-mining and ontologies: new approaches to knowledge discovery of microbial diversity.* May.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy, August. Association for Computational Linguistics.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Radim Rehurek. 2022. Gensim - pypi, url: `https://pypi.org/project/gensim/`.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Raymon van Dinter, Bedir Tekinerdogan, and Cagatay Catal. 2021. Automation of systematic literature reviews: A systematic literature review. *Information and Software Technology*, 136:106589.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

# Appendices

## A Methods, techniques, devices to be used in research

In this section, we describe methods and techniques used in developing the project solution.

### A.1 Keyword extraction

To detect the main concepts of the given documents, we decided to use Topic Modeling. It is an unsupervised machine learning method, that scans a set of documents and clusters them into groups represented by similar abstract topics. The conventional technique LDA (Blei et al., 2003) treats a document as a bag-of-words. Consequently, it loses the context and the order of the words. To prevent order loss and profit from the context of the given word, text embedding techniques have been used in various tasks. In recent years they became popular in the topic modeling field. Therefore, we decided to use BERTopic (Grootendorst, 2022b). However, we compare its performance with LDA.

#### A.1.1 LDA

The Latent Dirichlet Allocation is a generative probabilistic model for finding hidden topics in the given corpora, proposed in (Blei et al., 2003). It makes a few assumptions:

1. topics are the statistically significant words in given corpora,
2. documents are a mixture of topics,
3. topics are a mixture of words.

Based on them, LDA calculates the probability density of topics in the document.

Before performing the algorithm pre-processing is needed, words need to be tokenized and a number of expected topics need to be given (in this work it is going to be denoted as Q). After that, the word-document matrix is created. This matrix is then divided into two matrices: document-topic and topic-word one.

LDA is an iterative process. In the first iteration, the randomly selected topics are assigned to each word. After that, LDA tries to optimize the results. In order to do this, it examines each word separately. Assuming that all assigned topics, apart from the current one, are correct. LDA tries to find the best topic for a given word. To do this it calculates 2 probabilities;

1. p1: proportion of words in a given document with a given topic (q),
2. p2: proportion of the documents in which the word (w) has the topic q assigned.

Using those probabilities, it detects the most relevant topic for a given the word and reassigns it.

For each word in each document, the procedure is repeated until a steady solution is found. In the end, the list of Q tuples containing the topic number and the list of most informative terms with their probability is given. LDA does not interpret topics, this step needs to be performed manually (it only provides the topic number and the informative words, the user needs to add the topic description/ name if needed).

### A.1.2 BERTopic

In this project, we use the (Grootendorst, 2022a) BERTopic framework implementation.

To generate topic representation, BERTopic goes through 3 main steps.

First, it embeds documents in order to create their representation in vector space and compare their semantic meaning. As a default, it uses Sentence-BERT (SBERT) framework (Reimers and Gurevych, 2019a), which enables converting sentences into vector representation using a pre-trained language model. SBERT is an extension of the traditional BERT model, for which calculating the sentence probability is a very time-consuming task. As authors of the (Reimers and Gurevych, 2019b) claim, by adding the pooling operation at the output of the BERT, the time of finding the most similar sentence pair in a collection of 10000 sentences was reduced from 65 hours to 5 seconds. Therefore, the BERTopic framework, by default, makes use of the SBERT model. It also allows using other pre-trained sentence embedding or custom models.

Subsequently, it performs clustering. Due to high space dimensionality, calculating the distance might become ill-defined. Therefore, to reduce dimensionality UMAP (McInnes et al., 2018) algorithm is used. The reduced embeddings are clustered using HDBSCAN (McInnes et al., 2017). The BERTopic framework allows changing both dimensionality reduction and clustering algorithms. The aforementioned techniques are used as default methods.

The last step is finding the topic representation. As a default, the modified TF-IDF procedure is used. The original procedure combines term, and inverse document frequency:

$$W_{t,d} = tf_{f,d}\dot{l}og(\frac{N}{df_t}), \tag{4}$$

where $tf_{t,d}$ is the frequency of the term $t$ in document $d$, N is the number of documents and $df_t$ is a document frequency that shows how much information the term provided in the document. In BERTopic this procedure is generalized to clusters of documents. Firstly, all documents in the cluster are concatenated, and then TF-IDF is modified and obtained by the formula:

$$W_{t,c} = tf_{f,c}\dot{l}og(1 + \frac{A}{tf_t}), \tag{5}$$

where $tf_t, c$ is a frequency of the term $t$ in the class $c$. $C$ is concatenated into one document collection of the documents from the same cluster. $Tf_t$ is a class frequency, measuring how much information the term provides to a class. By using the modified TF-IDF formula the importance of the words in a cluster, rather then in the document, is modeled.

## A.2 Tagging tools

Keywords, extracted in the previous step, might incorporate different names to describe the same concepts. Therefore, to make use of them it is essential to perform mapping into existing ontologies. The ontology provides the standardized, homogenous, and informative concept, that describes the given keyword. The task of tagging the word with an entity existing in the ontology is called entity normalization, entity grounding, or entity categorization. As an example of the biomedical entity grounding, we will use the Onto-Biotope Ontology (Nédellec et al., 2018). Given the words ”pediatric”, ”respiratory” and ”children less than 2 years”, we aim to the appropriate tags in the ontology. For the first two words, the task is relatively simple. ”Pediatric” ought to be linked to ”pediatric patient” and ”respiratory” to the ”respiratory tract part”. Both of those examples are lexically similar. In the last case, the linking is not that trivial. ”Children less than 2 years” should be tagged as a ”pediatric patient”, even though the lexical similarity does not exist. The given example was derived from (Karadeniz and Özgür, 2019). Moreover, in the biomedical domain, the number of semantic categories is greater than the entities mentioned in available training data sets. For example, Onto-Biotope ontology consists of 2221 categories, while only 747 of them were mentioned in the training data set. Therefore, we decided to use unsupervised annotation techniques. In this section, we described two approaches tested in our solution. The first is NCBO tagger, which annotates data based mostly on direct string matching. The second, which uses word embeddings to link entities using word.

#### A.2.1 NCBO tagger

NCBO tagger (Jonquet et al., 2009) is a result of an initiative to construct a solution for annotating biomedical data with the use of a great number of ontologies. At the time of releasing the solution it used over 200 ontologies and this number is constantly increasing.

The way NCBO tagger works is simple, yet in many cases powerful enough. It uses a few steps to tag each token of an input free-text.

First of all, a direct string matching is performed. The dictionary of ontologies concepts is used for this purpose. It is constructed by pooling all concept names or other string forms (synonyms, labels) that syntactically identify concepts. Then, tokens from the input string are matched to this dictionary entries.

Second step is performed by *is_a transitive closure*, which aims to explore the relations in ontologies, namely for a given matched concept it searches its subsequent ancestors in the parent-child hierarchy and can match them as tags for a given token as well. The number of ancestors to look through is paramterizable.

Next, an *ontology-mapping component* tries to find relations between different ontologies, e.g., when a given concept is matched for a token, it can be linked to a respective concept in another ontology, and the ontology can also be traversed.

As a result, NCBO produces quite a lot of tags for each input text token. Our first trials showed that they are usually relevant, however, often to many of them is generated. Hence the need to select the most suitable tag and possibly perform disambiguation.

#### A.2.2 BIOBert

BERT (Vaswani et al., 2017) (Bidirectional Encoder Representations from Transformers) is a language processing model developed by Google that has been widely used for natural language processing (NLP) tasks such as text classification, language translation, and named entity recognition. It is based on the transformer architecture, which uses self-attention mechanisms to process input sequences in parallel, allowing the model to effectively handle long-range dependencies in language and achieve strong performance on a variety of NLP tasks.

BioBERT (**?**) is a variant of the BERT model that has been specifically designed for natural language processing (NLP) tasks in the biomedical domain. It was trained on a large dataset of biomedical literature and has been shown to perform well on a variety of NLP tasks in the biomedical domain.

#### A.2.3 Words embedding tagger

The approach mentioned in (Karadeniz and Özgür, 2019) is based on the assumption that semantically similar words have similar vectors in the embedded space.

Before computing the word embedding vectors, the preprocessing is performed. The words need to be free from stop words, and non-ASCII characters.

Next, the word vectors are calculated, using the pre-trained model. For multi-word entities, each word is transformed separately and the average vector is calculated. After the conversion of both tagged data and ontology concepts, their similarity is measured. We use cosine similarity which is calculated by the given equation:

$$cosine\_similarity = \frac{AB}{\parallel A \parallel \parallel B \parallel}, \qquad (6)$$

where A and B are the vectors. After performing those steps, the list of closest ontologies concepts is given.

### A.3 Words tags disambiguation

In free-text data, the same word can occur in different contexts and with different meanings. For example, if working with data containing information about wines, *Burgundy* can refer to the name of the wine or the region in France (Bindelli et al., 2008b). Hence, it should be decided whether to tag *Burgundy* with *Wine name* or *Country Region*. This information should be based on the context of a tagging word in an input text. If it comes to medical data, the term *blood pressure* can have three senses, namely *organism function*, *diagnostic procedure* and *laboratory or test result* (Alexopoulou et al., 2009).

We propose a method inspired by Alexopoulou et al. (2009). It is based on selecting the sense of a given word (or in general token) that is the closest to senses of other words appearing in the context. In the following subsections we describe the original Closest Sense method (sentence-based) and the modification than was incorporated in our solution.

### A.3.1 Sentence-based Closest Sense method

Let us suppose that we want to tag tokens in the sentence: *I also tracked lipid profiles, HBA1C, blood pressure, body mass index, hostility and nicotine use.* As mentioned above, *blood pressure* can have multiple senses since it is ambiguous - three tags are possible (assuming they are concepts of some ontology): *organism function*, *diagnostic procedure* and *laboratory or test result.*

To decide which tag should be assigned to *blood pressure*, we explore tags of other words appearing in the sentence. Let us assume that the senses of the occurring terms are *laboratory procedure* (lipid profile), *gene or genome* (HBA1C), *diagnostic procedure* (body mass index), *mental process* (hostility) and *organic chemical* (nicotine). Then for blood pressure, we choose the sense that is on average closer to the senses of the co-occurring terms than the other candidate senses.

What the *closeness* means can be treated in various ways. For example, semantic distances utilizing the ontologies (like subsumption distance or subtype-aware signature distance) can be used (Alexopoulou et al., 2009). The other way could be to incorporate words embedding and investigate cosine similarity.

### A.3.2 Keywords-based Closest Sense method

Since our task aims to tag keywords instead of particular words in free text, we plan to modify the Closest Sense algorithm.

First of all, for a given ambiguous keyword, we are going to treat other keywords extracted for a given text as the context, instead of words that occur in the same sentence.

Secondly, in the case of our problem, each keyword is ambiguous on a similar level (all keywords will have numerous candidate tags assigned). This is a difference in regards to what Alexopoulou et al. (2009) explored: they assumed only a given word in a sentence is ambiguous while other words have correctly assigned tags. That is why we propose the following iterative procedure: given a set $K$ of keywords $k_j, j = 1, \ldots, |K|$ for a given document and sets $|T_j|$ of candidate tags: $t_{j,i_j}$ for $j$-th kyeword, $i_j = 1, \ldots, |T_j|$ perform *max_iter* times:

1. Take subsequent keyword $k_j$ and assume this keyword is ambiguous, while all other keywords have correct tags assigned (take first tag in the candidate list for these keywords).

2. For each candidate tag $t_{j,i_j}$ calculate the similarity distances to other keywords tags and sort the list of tags from $|T_j|$ by decreasing similarity distance. As a result, at the top of the list we have the best tag for $k_j$ according to the current state.

3. Go to point 1. taking next keyword.

After $max\_iter$ iterations of above points, each $k_j$ will be considered $max\_iter$ times. This heuristic can help to choose keywords tags according to the context of the entire document.