UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Elizaveta Yankovskaya

Extraction and Classification of App Features from App Reviews

Master's Thesis (30 ECTS)

Supervisor: Sven Laur

Tartu 2017

Extraction and Classification of App Features from App Reviews

Abstract:

The number of tools for bioinformatics is constantly increasing. To organize the available information and to facilitate the search, different ontologies are used. Today annotation of new descriptions is done manually, which is time-consuming and not always correct. We proposed a new annotation method, which, based on the description of the tool, offers one or more annotation labels in accordance with the ontology. In our method, we applied modern methods of natural language processing, such as latent Dirichlet allocation and word2vec. We compared the manual annotation labels with the labels obtained by using our algorithm and the first results look auspicious.

Keywords:

natural language processing, text analysis, topic modeling, bioinformatics

CERCS:

P170 (Computer science, numerical analysis, systems, control),

B110 (Bioinformatics, medical informatics, biomathematics, biometrics)

Rakendusi klassifitseerivate tunnuste eraldamine nende kirjeldustest

Lühikokkuvõte:

Aasta aastalt on kasvanud bioinformaatikas kasutatavate rakenduste arv. Selle tulemusena on konkreetse ülesande lahendamiseks sobiliku rakenduse leidmine muutunud keerukaks ülesandeks. Rakenduste kirjelduste paremaks süstematiseerimiseks ja otsitavaks muutmiseks on kasutusele võetud erinevaid märksõnade ontoloogiaid. Hetkel annoteeritakse kirjeldusi käsitsi, mis on ajamahukas ning ei anna alati õigeid tulemusi. Antud töös kirjeldame uut annoteerimismeetodit, mis pakub automaatselt välja ühe või mitu märksõna kasutades selleks vaid tööriista vabatekstilist kirjeldust. Selleks kasutab meie meetod uusimaid loomuliku keele töötlemise meetodeid nagu Dirichlet' peitlahutus (latent Dirichlet allocation) ja sõnade vektoresitust (word2vec). Esmane võrdlus meie poolt välja pakutud algoritmi ja käsitsi saadud märgendusega näitab, et tulemused on paljulubavad.

Märksõnad:

loomuliku keele töötlus, tekstianalüüs, teemade modelleerimine, bioinformaatika

CERCS:

P170 (Arvutiteadus, arvanalüüs, süsteemid, juhtimine),

B110 (Bioinformaatika, meditsiiniinformaatika, biomatemaatika, biomeetrika)

Contents

1 Introduction

In recent years the number of web pages, documents, articles, reviews located on Internet has increased dramatically. For example, the number of articles in Wikipedia has increased from 1.5 M to 5.3 M for the last ten years [1]. One of the main issues faced by Internet users is how to find a needed resource. So, it would be useful if all information were analyzed and systematized automatically.

A number of unsupervised techniques has been proposed for the solving this problem. For example, it is possible to cluster documents in a hierarchical or flat (i.e., by using K-Means clustering) structures. The simplest way to characterize a document with a short description is to use document vectors with term frequency – inverse document frequency (tf-idf) scoring [2]. Later, a latent semantic analysis (LSA) was proposed [3]. This technique uses singular value decomposition (SVD) of term-document matrix. From LSA probabilistic latent semantic analysis (pLSA) [4] was evolved and a little bit later Latent Dirichlet allocation (LDA) [5] was developed. LDA and pLSA methods use a probabilistic method instead of using term-document matrices.

Speaking of a text classification, it is necessary to mention neuron networks. For example, character-level convolutional networks [6] have a great advantage compared with other techniques: they allow to work with languages where tokenization into words is not possible.

1.1 The aim of the thesis

At this moment there are tremendous number of bioinformatics tools. So sometimes researchers cannot find the required tool as fast as they want. For this reason it is a good idea to have a brief systematized description of all tools in one place. Today there are several such online database, for example, *bio.tools* [7] and *omictools* [8].

However, tools in these databases are annotated manually. Firstly, it is a time-consuming process. Secondly, people who annotated these tools may make mistakes for different reasons, for example, due to insufficient knowledge of the classifier. For

these reasons, it would be better if the manual annotation process was replaced by an automatically classifier.

Last year a master student Erik Jaaniso under supervision Hedi Peterson developed a program [9] that automatically annotates bioinformatics tools according to EDAM ontology [10] based on free description texts. They proposed an annotation method that is based on calculating inverse document frequency weights of words from description texts and ontology terms and computing matching between them. Also, the presented method gives different weights to words coming from a short description of the bioinformatic tool, keywords, a detailed description, etc. As a result, they have got promising results.

The purpose of this work is to apply other methods that could improve the quality of automatic annotation. As the basis of our approach we have chosen Latent Dirichlet Allocation as one of the popular topic modelling techniques now.

1.2 Related works

Latent Dirichlet Allocation is applied to retrieve information from the texts. This method allows to discover hidden topics in documents and can be successfully used in sentimental analysis, clustering and classification tasks. Since in this work we want to classify the bioinformatics tools rather than get their sentiment score, then in the future we will only discuss the classification and clustering problems and describe below a few works that use LDA to solve these tasks.

So, in the article [11] the authors showed that the LDA-based algorithm for software categorization gave results comparable to the current categorization algorithm. After parsing software systems and applying LDA to them, they combined similar topics into the same categories. To do that, they calculated cosine similarity between each pair of topics, and if it is greater than 0.8, topics were grouped into one category. After that, authors assigned software systems to the obtained categories by using topics probability computed by applying LDA.

Researchers [12] proposed a method similar to our approach. They used not only topic information features derived from LDA, but also added term frequency features

to them. After combined two types of features they used received vectors as features for the support vector machine. Due to this procedure they have got more precise results than using only term frequency features or topic information features.

In the article [13] authors applied LDA to set of legal judgments. After that they calculated cosine similarity between each document and the obtained topics. Based on the calculated similarity, they put each document to a cluster. Authors got reasonable clusters, however there is a need to mention that they knew the "ideal" number of clusters.

1.3 The structure of the thesis

This thesis has the following structure:

- the second chapter provides a brief description of the main methods which we use to build our model;
- the third chapter introduces a pipeline of the proposed model, describes data that we use for solving of our task and gives a whole description of the experiments steps;
- the fourth chapter provides the obtained results and possible improvements of the proposed model.

2 Theoretical background

2.1 Term frequency – inverse document frequency

Term frequency – inverse document frequency (tf-idf) is a statistical measure used to figure out the importance of word inside a document. Intuitively, if a word appears in a text frequently, it might be substantial. However, if the same word appears in many documents, it does not give useful information about a document. For example, if we have corpus of texts about different animals, a word "cat" which appears many times only in one document will be important, but if all texts about cats, a word "cat" will be in almost all documents and will not provide some valuable information.

So, computing of tf-idf consists of three parts, which are described below.

1. Computing term frequency (tf):

$$tf(t) = \frac{n_t}{\sum_k n_k},$$

where n_t is number of times term t appears in a document and

$\sum_k n_k$ is total number of terms in the document.

2. Computing inverse document frequency (idf):

$$idf(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right),$$

where $|D|$ is total number of documents in the corpus and

$|\{d \in D : t \in d\}|$ is number of documents with term t in it.

3. Computing tf-idf:

$$tf\text{-}idf(t) = tf(t) \cdot idf(t, D).$$

Term frequency – inverse document frequency weighting is commonly used in information retrieval tasks as a basis for more complicated algorithms [14, 15, 16].

2.2 Probabilistic Topic Modeling

One of the ways to automatically organize and understand massive document collections is a probabilistic topic modeling. The main idea of probabilistic topic modeling is automatically to detect topics in documents. Topic modelling algorithms are set of statistical methods that analyze the words of the given texts and discover topics hidden in these texts. Any prior annotations or labelling of the texts are not required for topic modelling algorithms [17].

The most two common techniques of a probabilistic topic modeling are Latent Dirichlet Allocation (LDA) [5] and probabilistic Latent Semantic Analysis (pLSA) [4]. The both methods treat topics as word distributions and documents represents a mixture of corpus-wide topics. The main difference between them is that pLSA method does not make any assumptions about a prior topic distribution over documents and a prior word distribution over topics whereas in LDA the topic distribution has a Dirichlet prior on the per document topic and on the per topic word distributions.

2.2.1 The model of Latent Dirichlet Allocation

The LDA model that was proposed by David Blei, Andrew Ng and Michael I. Jordan [5] has three assumptions:

- the order of the words in a document does is irrelevant ("bag of words"),
- the order of the documents is irrelevant,
- the number of topics is known and permanent.

With these assumptions, we can define LDA model with the following notation. The number of all topics is K and ϕ_i is a word distribution in the topic i and $i \in \{1, ..., K\}$. The number of all documents is M and Θ_j is a topic distribution in document j and $j \in \{1, ...,M\}$. The number of words in a document is N and the topic assignment for the j-th document is Z_j, then $Z_{j,t}$ is the topic assignment for

the t-th word in the document j, where $t \in \{1, ..., N\}$. Finally, $W_{j,t}$ is the t-th word in the j-th document.

According to this notation, the total probability of the model is:

$$P(W, Z, \Theta, \phi) = \prod_{i=1}^{K} P(\phi_i) \prod_{j=1}^{M} P(\Theta_j) \prod_{t=1}^{N} P(Z_{j,t}|\Theta_j) P(W_{j,t}|\phi, Z_{j,t}), \tag{1}$$

$$\Theta \sim Dir(\alpha), \tag{2}$$

$$\phi \sim Dir(\beta), \tag{3}$$

where α and β are hyperparameters of the Dirichlet distribution.

Hyperparameters of the Dirichlet distribution α and β in the equations (2) and (3) are less than one, as we assume that each document has a small subset of important topics and each topic has few keywords. If we set α and β to more than one, we would get that all topics and all keywords have almost the same probability.

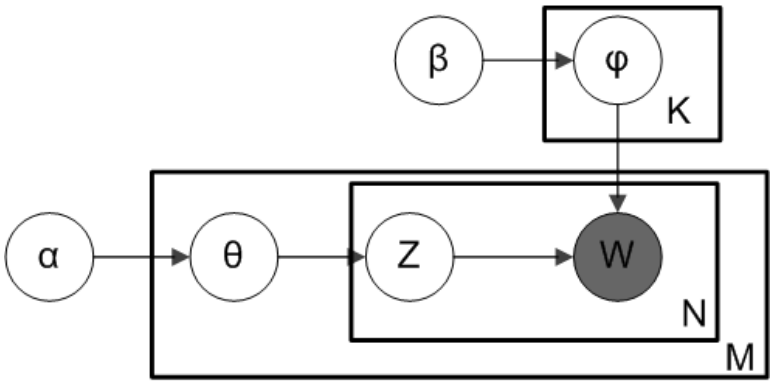

Figure 1: Graphical smoothed LDA model representation [18]. The grey circle represents observed nodes and the white circles represents the hidden nodes.

The visual scheme given above is a good interpretation of the equation (1). The first product in the formula (1) corresponds a topic plate, the third one corresponds a word plate and the second and the third products correspond to a document plate.

2.2.2 The algorithm of Latent Dirichlet Allocation

Latent Dirichlet allocation model can works with various learning algorithms, such as maximum a posteriori estimation, collapsed Gibbs sampling and Bayesian variational inference. According to the article [19], Bayesian variational inference is the best choice in terms of speed and efficiency.

Below we describe how to find latent topics in a corpus of documents less formally [20].

1. Define a number of topics (K). To find an optimal number of topics is a not simple issue. It is possible to find an optimal number of topics by using prior knowledge or trial-and-error method. Also, some estimation metrics exist, for example, perplexity. However, they do not always give human interpretable results [21].

2. Go through each document the algorithm assigns every word to one of the K topics according to a Dirichlet distribution. Due to a Dirichlet distribution if a word occurs several times it may be assigned to different topics. Topic assignments are temporary and will be improved in the next step.

3. Go through each word in each document the algorithm updates topic assignments computing two things:

   - how widespread is that word over all topics in all documents?
   - how many words in this document are assigned to the topic t?

   In this step, the algorithm assumes that all topic assignments but the current word are appropriate.

   After repeating this step multiple times, the stable state is reached.

Assume, we have two documents A and B and we want to discover topics that these documents contain.

A: Dinosaurs are a group of reptiles. It is unlikely to meet dinosaur today.

B: Walt Disney is an animation studio. It produced an adventure film "Dinosaur".

For example, we decided that our documents A and B have two topics and after the second step each word has a topic assignment, and we would like to update a topic assignment for a word "*dinosaur*" in the document A:

Document A	Topic assignment
dinosaur	???
group	topic 1
reptile	topic 1
unlikely	topic 2
meet	topic 2
dinosaur	topic 1
today	topic 2

Document B	Topic assignment
Walt Disney	topic 2
animation	topic 2
studio	topic 2
produce	topic 1
adventure	topic 2
film	topic 2
dinosaur	topic 1

According to the third step, there is a need to answer to two questions.

- How widespread is that word over all topics in all documents: a word "*dinosaur*" appears in both document and is assigned only to topic *1*. Thus, it is more likely that the word "*dinosaur*" belongs to topic *1*.

Document A	Topic assignment
dinosaur	???
group	topic 1
reptile	topic 1
unlikely	topic 2
meet	topic 2
dinosaur	**topic 1**
today	topic 2

Document B	Topic assignment
Walt Disney	topic 2
animation	topic 2
studio	topic 2
produce	topic 1
adventure	topic 2
film	topic 2
dinosaur	**topic 1**

- How many words in this document are assigned to the topic t: half of words in the document A is assigned to the first topic and the half to the second topic. The word "*dinosaur*" may be assigned equally likely to either topic.

Document A	Topic assignment
dinosaur	???
group	**topic 1**
reptile	**topic 1**
unlikely	*topic 2*
meet	*topic 2*
dinosaur	**topic 1**
today	*topic 2*

Document B	Topic assignment
Walt Disney	topic 2
animation	topic 2
studio	topic 2
produce	topic 1
adventure	topic 2
film	topic 2
dinosaur	topic 1

According to the results of this two criteria, the word "*dinosaur*" is reassigned to the first topic.

As a result, we have got the following distribution topics in the documents:

Document A	Topic assignment
dinosaur	topic 1
group	topic 1
reptile	topic 1
unlikely	topic 2
meet	topic 2
dinosaur	topic 1
today	topic 2

Document B	Topic assignment
Walt Disney	topic 2
animation	topic 2
studio	topic 2
produce	topic 1
adventure	topic 2
film	topic 2
dinosaur	topic 1

We can write it in another way:

A: 57% the first topic and 43% the second topic

B: 29% the first topic and 71% the second topic

So, we can suppose that the document A might be about biology and the document B about films.

2.3 Word2vec

Words in any text are related semantically, so if we represent them as separate elements, for example, a word " pterosaurs" may be depicted as "001" and "pelycosaurs" as "100", then we will get a little information about the text. One way to overcome it is to use vector representation of words. Vector space models treat words in a continuous vector space, where semantically similar words are located close to each other. All vector space models can be divided into two groups: context-count and context-predictive.

Context-count algorithms (e.g. tf-idf, LSA, LDA) compute how often a word appears with other words and map the obtained statistics to a vector. Context-predictive models (e.g. neural probabilistic language models like GloVe [22], Collobert and Weston model [23], word2vec [24]) try to predict a word from its surrounding words in terms of embedding vectors [25].

Word2vec is a set of neural network's algorithms that one of the most popular of the context-predictive models today. The main idea of word2vec is to maximize the cosine similarity between the vectors of words which appear in the similar context and minimize the cosine similarity of the vectors for words that do not appear close together. Word2vec takes as its inputs a corpus of text (for better results there is a need an enormous amount of documents) and produces a matrix of word-vectors, where each row is a unique word and a vector corresponding to this word [26, 27].

There are two models: CBOW (continuous bag of words) and skip-gram. The architectures of both models are shown in Figure 2.

CBOW takes as input $[w_{t-n}...w_{t-1}, w_{t+1}...w_{t+n}]$ and produces a target word w_t. Thus, it predicts the target word given the context. The value of n depends on the chosen window size. Skip-gram model works in the inverse way: it predicts the context $[w_{t-n}...w_{t-1}, w_{t+1}...w_{t+n}]$ given the word w_t [24]. For example, we have a sentence "my cats and dogs are friends", a CBOW model predicts a word "dogs" from context words ["my", "cats", "and", "are", "friends"] while a skip-gram predicts surroundings words ["my", "cats", "and", "are", "friends"] for a word "dogs".

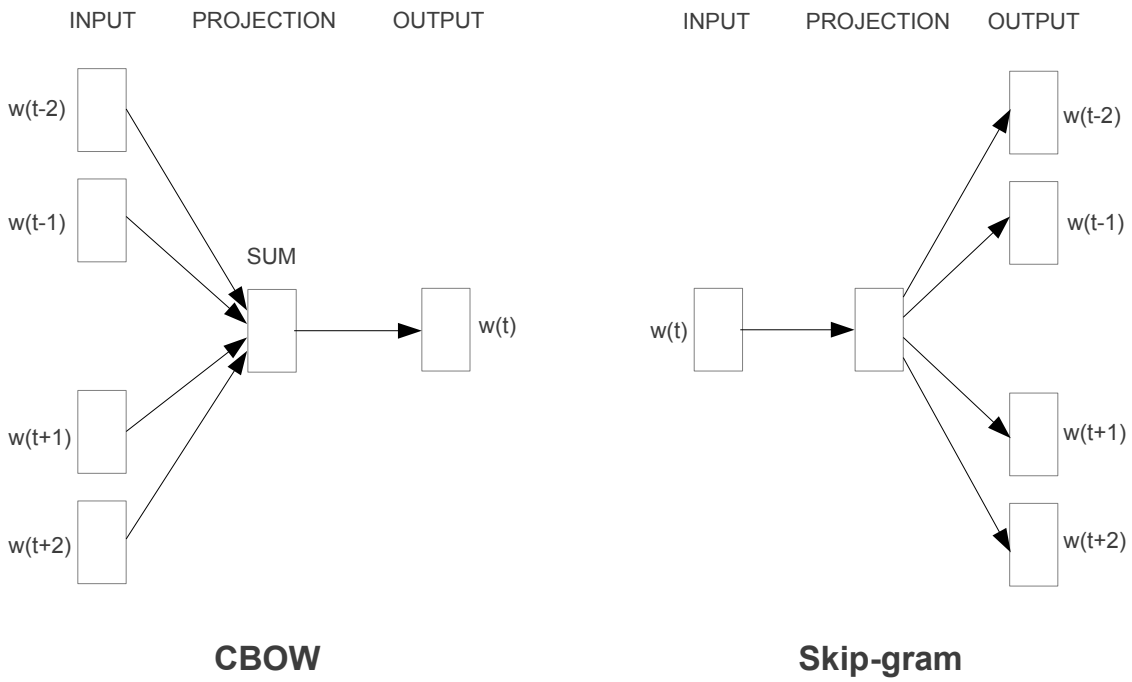

Figure 2: Model architectures: CBOW and skip-gram [24]

3 Experiment

Figure 3 depicts the data processing pipeline of the experiment. The blue square boxes illustrate steps along the pipeline and the yellow ellipses represent the essential intermediate outputs. In this section we will provide the detailed explanation of the experiment steps.

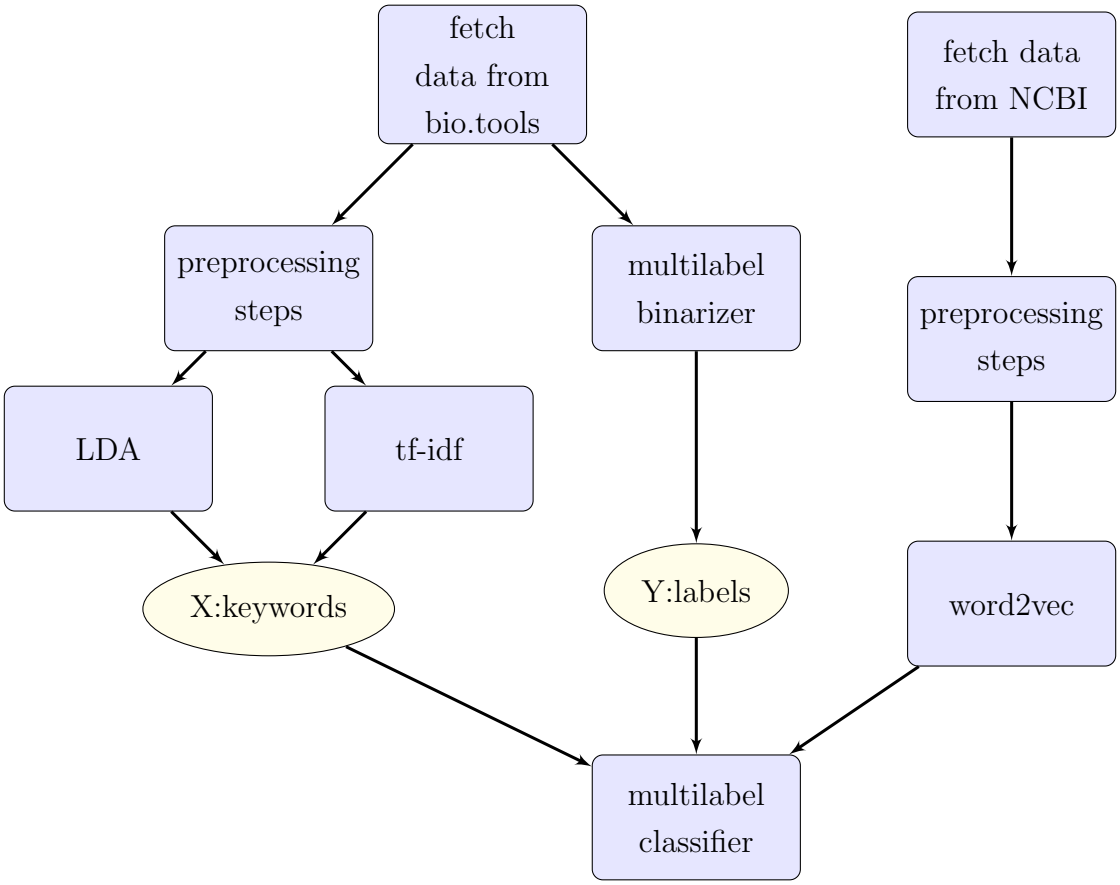

Figure 3: Pipeline of the experiment

3.1 Getting data and data set description

The first step in our work is getting the data. According to the pipeline of our experiment, we need data for training LDA model, word2vec model and then multilabel classification. Before describing the data, we will explain what EDAM ontology is.

3.1.1 EDAM ontology

EDAM ontology is a simply ontology of bioinformatics concepts [28]. It includes four main categories and one sub-category as defined in [10].

Topic:

> "A category denoting a rather broad domain or field of interest, of study, application, work, data, or technology. Topics have no clearly defined borders between each other"

Operation:

> "A function that processes a set of inputs and results in a set of outputs, or associates arguments (inputs) with values (outputs)"

Data:

> "Information, represented in an information artifact (data record) that is 'understandable' by dedicated computational tools that can use the data as input or produce it as output"

Format:

> "A defined way or layout of representing and structuring data in a computer file, blob, string, message, or elsewhere"

Data ⇒ Identifier:

> "A text token, number or something else which identifies an entity, but which may not be persistent (stable) or unique (the same identifier may identify multiple things)"

Figure 4 shows the classification mentioned above and also presents examples from each category. For instance, the category *Topic* may be phylogenetics and/or transcriptomics.

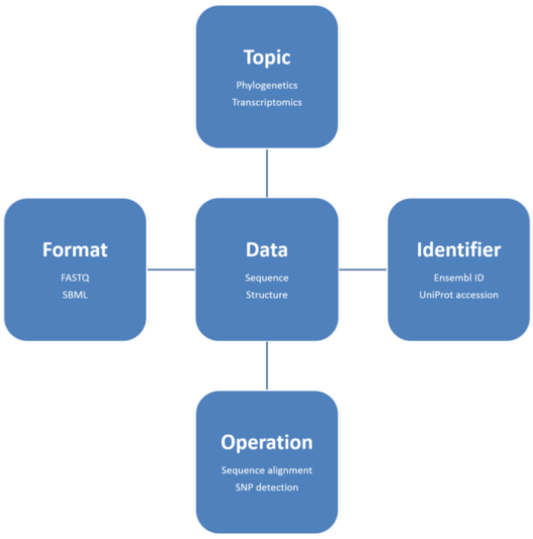

Figure 4: EDAM concepts [10]

Mostly, the *Data*, *Format*, and *Operation* categories consist of concepts strictly in the field of bioinformatics and computational biology whereas concepts purely concerning computer science, biology, etc., are not included. The *Topic* category contains wide-ranging interdisciplinary concepts from the biological and medical domains [10].

3.1.2 Data for LDA model and multilabel prediction

The web service *bio.tools* [7] is one of the main databases of bioinformatics tools. It provides the information about authors and versions, a small description, links to the essential publications (which, as a rule, give a complete description of the tool) and classification topics that are corresponds to EDAM classification system.

For example, a tool *Kaiju* has a following EDAM concepts:

topic	operation	inputs	outputs
Metagenomics	Taxonomic classification	Nucleic acid sequence (raw) (FASTQ, FASTA)	Taxonomy (TSV)

For our goal there is a need to extract a text of publications and corresponding classification topics. We used classification topics as reference values (labels) of multilabel classification and text of publications to train LDA model.

During analysis of classification topics we faced the following problem. Registered users can add their tool to bio.tools and describe their functionally based on EDAM ontology. Since ontology is a hierarchical system, some users select the upper level, the other users — the deepest sub-level, while others select the entire chain of levels from the top to the lowest.

For example, as shown in Figure 5 "visualisation" is a high level in *Operation* category and it has many sub-levels, one of them is "sequence visualisation" and it in turn consists of three sub-sub-levels. So, some users chose only "visualisation" as an operation topic, other decided to pick "sequence visualisation" and someone else selected "dotplot plotting" and "sequence visualisation".

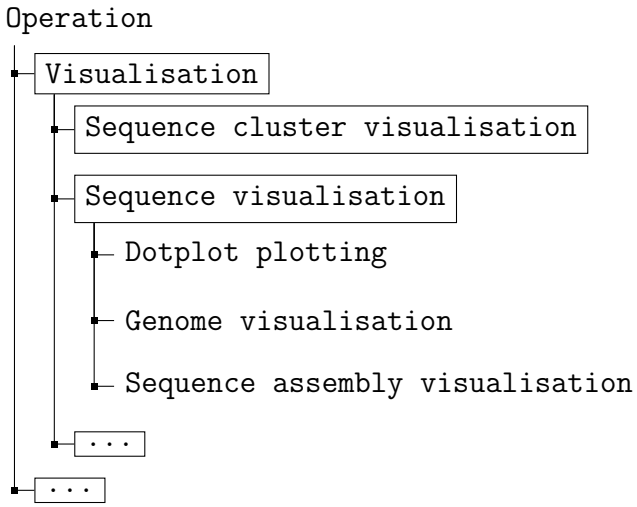

Figure 5: Example of EDAM structure

To remove this non-uniformity we decided to keep only the top and the first sub-level of topics. For the example mentioned above, we kept only "visualisation" (the top level), "sequence cluster visualisation" and "sequence visualisation" (in Figure 5 the remained topics are placed in boxes).

To test the performance of our method we kept only *Operation* category. After our modification of EDAM structure we got 20 different topics.

To extract texts of publications we used an edam-mapper script[1]. We gathered only these descriptions which have open access.

We collected 3269 descriptions. As can be seen in Figure 6 some of descriptions are very short. Since Latent Dirichlet Allocation has proven to be successful when applied to long texts and has given incoherent topics when applied to short texts [29], we decided to keep only relatively long texts. After extensive experiments, we discovered that a text length should be 500 or more words after preprocessing. As a result, we got 2368 descriptions.

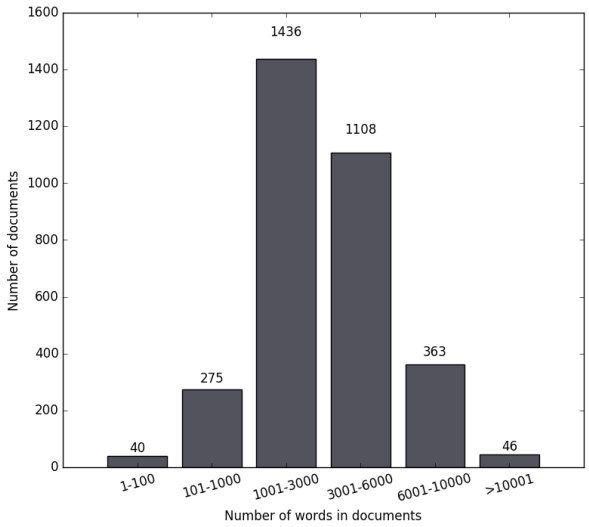

Figure 6: Distribution of words over texts before preprocessing steps

[1]https://github.com/edamontology/edammap

3.1.3 Data for word2vec model

To get stable results for word2vec there is a need a huge amount of data. After gathering information from *bio.tools* we have a database with size about 127 Mb (the number of uniqie words is 400,128.) Since this is not enough for learning the word2vec model, we need to get more data. Our data obtained from *bio.tools* is related to bioinformatics. So, we need to get more data related to bioinformatics. One of ways to solve this issue is to use databases of the National Center for Biotechnology Information (NCBI).

The National Center for Biotechnology Information (NCBI) is part of the United States National Library of Medicine. It contains a series of databases relate to biotechnology and biomedicine. All these databases are available through the *Entrez* search engine [30].

To train our word2vec model we used data from *PubMed*[2] and *PubMed Central*[3] databases. *PubMed* database is the MEDLINE database of abstracts on biomedical, bioinformatics and life science topics. *Pubmed Central (PMC)* is a free digital repository that contains archives of publicly accessible full-text articles of biomedical and life sciences journals. To get data we used the *Entrez* search engine looking for keywords "bioinformatics", "taxonomy" and "genome".

As a result, we obtained 418,190 abstracts and 79,955 articles. The size of the obtained dataset is about 3.7 Gb. After removing stop words, like *the*, *and*, *but*, we received 758,690 unique words.

3.2 Preprocessing

Before applying any model to the collection of documents, there is a need to make preprocessing steps. Below we will describe the steps that we consider necessary for our corpus of texts.

Useless parts of text removal

We removed all words before a word "abstract" and after words "references", "ac-

[2]https://www.ncbi.nlm.nih.gov/pubmed/
[3]https://www.ncbi.nlm.nih.gov/pmc/

knowledgement", "competing interests", "authors contributions", "contribute support" as they do not contain useful information about a described tool.

Select sentences carried "useful" information

To select sentences which have useful "payload", firstly we split texts into sentences by using *nltk.tokenize.punkt* [31] and then applied two approaches.

The first approach is simple: we extracted only those sentences which contain words related to biology, software and/or experiment.

For example, words such as "gene", "dna", "intron", "exon" related to biology, "algorithm", "software", "platform" to software and "sensitivity", "specificity", "accuracy" to experiment.

To create these lists of words, we took the words which we associated with biology, experiment and software. We also applied *K-means* [32] clustering to the whole corpus of texts (by using the tf-idf matrix and cosine similarity) and took top 25 words that are nearest to the cluster centroid and are related to biology, experiment and software.

The second approach is a little bit sophisticated: we calculated tf-idf weights for each word and then chose only those sentences whose normalized tf-idf score (normalized tf-idf score is sum of tf-idf score of each word in a sentence divided by number of words in the sentence) is more than 0.8.

After initial computations the second approach showed the better performance than the first one. So we decided to use the second approach in the future.

Punctuation and web links removal

Before removing of punctuation we replaced all words with hyphens by the same words without hyphens, for example, a word "inc- reasing" is replaced by "increasing" or "k-mer" was replaced by "kmer". In some cases, like "Z-score", it works incorrect (a word "Zscore" will be created); however, we assumed that these cases are uncommon compared to justifiable cases.

After that, we removed all punctuation. As with the removal of the hyphen, deleting

the entire punctuation can lead to the wrong result. For example, the "superfamily/-family concept" will be replaced by a "superfamilyfamily concept", however, these situations are rare.

We also cleared away web links and e-mail addresses as they do not provide information about a bioinformatics tool.

Tokenization to words

We replaced the input text by list of words (tokens) by using *nltk.tokenize* [33]. We kept the words which contain at least two letters. For example, a word "r" and a word "r45" are removed because they include only one letter whereas a word "bi" is held. Also, we saved only letters from words, for instance, a word "clark31" is saved as "clark".

Deleting numbers can again lead to incorrect situations, for example, instead of strains "JS614" and "JS666" we obtained one word "JS". However, we assumed that keeping the digits in words will not lead to an improvement in the LDA model's performance, while deleting the numbers will reduce the number of keywords describing the topic.

Stopwords removal

We removed words that are very frequent in English, like "the", "and", "are" and so on. We used the standard set of words from the library *stop_words* [34].

We also added to this set some common words appeared in texts and not given useful information: "doi", "figure", "et", "ii", "iii", "fig", "none".

Lemmatization

There are two main approaches to reduce inflectional forms of words: stemming and lemmatization. Stemming reduces words to stems while lemmatization reduces them to lemma. For example, a word "sequencing" has a stem "sequenc" and two lemmas "sequencing" and "sequence" depending on the morphological analysis of the word [35]. We decided to use lemmatization to keep morphological differences as we consider that it is important for our method.

We applied *nltk.stem.wordnet* [36] to find the different inflectional forms of a word and to get the lemma for a word. For example, the verb "to know" may appear as "know", "knows", "knew", "known" and its base form or lemma is "know".

Thereby, using this technique we can reduce the number of possible keywords describing a topic.

Search of collocations
To find collocations like "gene expression", "amino acid", "protein protein interaction" we used the *NLTK* packages *BigramCollocationFinder* and *TrigramCollocationFinder* [37].

We kept collocations which appear at least twice in one document and at least nine times in all documents. After that we replaced discovered collocations by one term, for example, a collocation "gene expression" was transformed to "gene_expression".

Removal all words except nouns, verbs and adjectives
The keeping or removal of certain parts of speech is a disputable issue. On the one hand, we reduce the number of possible words describing the topic, on the other hand, we can lose important information.

We used a *NLTK* module *nltk.tag.stanford* [38] for communicating with the *Stanford taggers* [39] to determine the parts of speech. For our corpus of texts, we decided to keep nouns, verbs and adjectives and remove adverbs, conjunction prepositions and other parts of speech, since they usually do not provide essential information about a bioinformatic tool.

3.3 Getting keywords using Latent Dirichlet Allocation

We applied Latent Dirichlet Allocation implementation developed by *sklearn* [40] to our data. We used default values of prior of document topic distribution and prior of word topic distribution. Both are equalled $\frac{1}{n}$, where n is number of topics. Varying values of prior of document topic distribution and prior of word topic distribution and inspecting the behavior of findings with different values is left as future work. We varied a parameter "number of topics" between 8 and 50.

We applied our LDA model to the whole data set. As an example, the obtained ten keywords for each topic ("number of topics" is 11) is presented in Table 1.

Table 1: Extracted topics with keywords and interpretation of topics

Topic	Keywords	Topic interpretation
1	model, data, value, method, sample, number, distribution, cluster, result, test	analysis, clustering
2	gene, pathway, network, interaction, set, disease, analysis, list, expression, cancer	disease pathways
3	database, data, information, search, provide, annotation, provide, include, query, link, resource	data and query
4	sequence, alignment, match, score, algorithm, tree, align, search, program, multiple	sequence alignment
5	gene, genome, sequence, species, cluster, annotation, transcript, genomic, human, exon	genome and sequence annotation
6	sequence, rna, target, structure, dna, mutation, nucleotide, design, position, secondary	sequence structure alignment
7	motif, site, region, sequence, promoter, regulatory, binding, pattern, identify, element	promoters, transcription factor binding sites
8	structure, protein, residue, pdb, structural, model, atom, molecule, ligand, interaction	protein interaction
9	protein, prediction, peptide, predict, domain, method, sequence, amino acid, score, performance	prediction
10	user, data, tool, analysis, file, provide, format, software, feature, result	software
11	read, sequencing, genome, snp, assembly, sequence, variant, data, number, coverage	sequencing

Table 1 shows a possible interpretation of topics keywords. Some of topics in-

terpretations describe a broad area, for example, "data and query", while others represent rather narrow area like "transcription factor binding sites".

The obtained keywords and their corresponding mapping to EDAM topics are shown in Table 2. As can be seen, the same set of keywords is matched by different terms from the ontology. To resolve this problem, we could take more than one the assigned topic. However, for a vast collection of documents and corresponding sets of reference values, it is likely that the same problem will be appeared again. The reason of the described issue might be that LDA allows to find topics that described global features of objects rather than local ones [41]. Another possible reason may be that LDA forms different topics than EDAM ontology. One more possible reason is the incorrect classification of tools on *bio.tools*.

Table 2: Document titles, keywords of the top topics assigned to documents and reference values

Keywords of the top topic assigned to documents	**EDAM topics**	**EDAM operation topics**
3DLigandSite: predicting ligand-binding sites using similar structures		
structure, protein, residue, pdb, structural, model, atom, molecule, ligand, interaction	structure prediction, protein binding sites, nucleic acid sites, features and motifs, small molecules, protein structure analysis	prediction and recognition, structure prediction, protein-ligand docking, protein binding site prediction, protein structure prediction
3D-partner: a web server to infer interacting partners and binding models		
structure, protein, residue, pdb, structural, model, atom, molecule, ligand, interaction	database management, proteins, protein binding sites, structure analysis, sequence composition, complexity and repeats	database comparison, protein comparison, comparison, protein model validation, molecular docking

3.4 Getting keywords using tf-idf

If the reason for having of the same data sets for different EDAM topics is that LDA does not discover local features, then we can use tf-idf to find aspects (local features) of the object [42].

We obtained terms with tf-idf weights by using *TfidfVectorizer* [43] from *sklearn*. Almost all parameters were used by default except of "ngram_range". We set it in (1, 3) to find not only unigrams but also bigrams and trigrams.

We kept only those unigrams, bigrams and trigrams whose weight was above the specified threshold (0.1).

Table 3: Document titles, keywords of the top topic obtained by LDA and keywords obtained by tf-idf and reference values

Keywords of the top topic assigned to documents	**EDAM topics**	**EDAM operation topics**
3DLigandSite: predicting ligand-binding sites using similar structures		
structure, protein, residue, pdb, structural, model, atom, molecule, ligand, interaction, *dligandsite, ligand, prediction, residue, conservation, use, cluster, cite, bind site*	structure prediction, protein binding sites, nucleic acid sites, features and motifs, small molecules, protein structure analysis	prediction and recognition, structure prediction, protein-ligand docking, protein binding site prediction, protein structure prediction
3D-partner: a web server to infer interacting partners and binding models		
structure, protein, residue, pdb, structural, model, atom, molecule, ligand, interaction, *partner, interact, residue, template, score function, domain, synthase, score, hydrogen bond, server, query, impala, energy*	database management, proteins, protein binding sites, structure analysis, sequence composition, complexity and repeats	database comparison, protein comparison, comparison, protein model validation, molecular docking

We combined keywords obtained by applying *TfidfVectorizer* and keywords of the top topic retrieved by using LDA. The example of a new set is shown in Table 3 (we italicized tf-idf words).

As can be seen, now documents have different set of keywords. Thus, this solution allows us to create unique sets of words for each document.

3.5 Vectorizers based on word2vec

We applied *gensim* [44] realization of *word2vec*. We set the following parameters: model is CBOW, the maximum distance between the current and predicted word within a sentence is ten, the dimensionality of the feature vectors is 300.

Our goal of using *word2vec* was to get vectors of terms that we will use to build features. There are several ways to do it.

For example, we have a corpus of texts and one of the texts is "taxonomic classification's program". After applying *word2vec* we obtained the following dictionary:

term	word2vec's vector	tf-idf
program	[... 0.3 -0.05 0.2 ...]	1.5
taxonomic	[... -0.2 1 -0.9 ...]	6
classification	[... -0.4 -0.1 -0.3 ...]	2

The easiest way is to calculate the average feature vector for each text. For the example above, it will be [... -0.1 0.28 -0.33 ...]. However, according the initial results this way showed the worst performance, so we decided not to use it in the future.

The another way is to multiply word vectors with their tf-idf scores and then take the average of them. The resulting vector for the example will be [... -0.5 1.9 -1.9 ...].

The third way is to apply a vector quantization. By using clustering algorithms, all obtained vectors are split into baskets ("clusters") having roughly the equal number of vectors closest to them. The final vector is a vector with a number of features equal to the number of clusters and a feature is how many terms of each text are contained in each cluster. Thus, we use semantically related baskets instead of individual terms. For example, we have five clusters and a term "program" corresponds

to the first cluster, “taxonomic” is contained in the third cluster and “classification” in the fourth cluster. The resulting vector will be [1 0 1 1 0].

To perform the third way we used *K-Means* [32] from *sklearn* library. We varied the number of clusters from 5000 to 15,000 with step 2500 and according the initial results we got the best performance with 10,000 number of clusters. We used this number of clusters for further calculations.

3.6 Multilabel classification

As in our task one set of keywords may be associated with multiple EDAM terms, we should use a multilabel classification. To build a multilabel classifier we used a module *multiclass* [45] from a *sklearn* library. It is developed to solve multiclass and multilabel problems. The most common approach is the binary relevance method: a multilabel problem is split into multiple binary classification problems, so we have one problem or classifier for each label, and each binary classifier is fitted to predict the relevance of one of the labels [46, 47]. A module *one-vs-the-rest* [45] from a *sklearn* library is developed to solved this issue.

To represent our multiple terms from EDAM as a vector we used *MultilabelBinarizer* [48]. For example, we have labels A, B, and C and samples with these labels [[A, B], [B], [A, B, C]]. After transformation an array looks [[1, 1, 0], [0, 1, 0], [1, 1, 1]] and it represents that the first sample has the labels A and B, the second sample has a label B and the third sample all of them.

As a prediction algorithm we applied the widely used classifier *SVM* [49]. We used *K-fold cross-validation* [50] from a *sklearn* library with $K = 10$ without shuffling to train our model. After all preprocessing steps we have 2368 texts, thus about 2131 texts are used as training data and 237 texts are used as a test set. We obtained 10 results for each metric (we wil discuss metrics in Section 4.2) and took the average of all results to get the final results.

4 Results

4.1 Parameters of models

Before we started to work with a whole data (3269 texts), we conducted a series of extensive experiments on a small corpus of texts (20% of the whole data) to find the main parameters that may be important for obtaining good results and to get a reasonable range of these parameters. After all the discussions and preliminary studies, we decided to choose the following parameters.

Number of top assigned topics from LDA (we discussed LDA in Section 3.3)

We chose for further experiments the number of top assigned topics: one, two, three, five and also only those topics which have weight more than 0.15 (further we will denote it as a "threshold").

During the preliminary calculations we took a larger number of topics, but the increase in topics led to worse results. This can be explained by the fact that usually each document has a small number of important topics, for example, the most texts has only two or three topics with weight more than 0.1.

Number of keywords assigned to each topic (we discussed LDA in Section 3.3)

We took 10, 25 and 50 top keywords assigned to each topic. We experimented with fewer words (five) and with more words (75) and better results were obtained with a number of words between 10 and 50.

Keywords from tf-idf

As we wrote in Sections 3.3 and 3.4, we want to use tf-idf keywords to solve the situation, when the same keywords obtained from LDA describe different topics. Although initial experiments showed that our approach is reasonable we decided to check it on the whole data.

Vectorizer

To transform documents to feature vectors we decided to use one built-in vectorizer: *CountVectorizer* from *sklearn* [43] ("CountV") and two custom vectorizers: TfidfVectorizer ("TfidfV") and ClusterVectorizer ("ClustV") based on word2vec (we defined both vectorizers in Section 3.5).

According to our preliminary calculations, a built-in *Tfidf Vectorizer* from *sklearn* [43] and a custom vectorizer that calculates the average feature vector for each text (we defined this vectorizer in Section 3.5) showed the worst performance.

4.2 Performance metrics

To evaluate our models defined in Section 4.1 we used accuracy, recall and precision [51]. Firstly, we calculated the metrics of a document d in the multilabel setting:

$$Acc(d) = \frac{|T \cap U|}{|T \cup U|},$$
$$Rec(d) = \frac{|T \cap U|}{|T|},$$
$$Prec(d) = \frac{|T \cap U|}{|U|},$$

where $|T|$ is the true set of labels and $|U|$ is the predicted set of labels.

After that, we computed accuracy for a test set:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} Acc(d_i),$$
$$Recall = \frac{1}{n}\sum_{i=1}^{n} Rec(d_i),$$
$$Precision = \frac{1}{n}\sum_{i=1}^{n} Prec(d_i),$$

where n is the number of documents in test set.

4.3 Results

Table 4 shows accuracy for all models when a training set consists of keywords obtained from LDA and tf-idf and when a training set consists of keywords obtained only from LDA. In this table we provided only average accuracy obtained by using 10-fold cross-validation for all range of topics.

Table 4: Accuracy of models with different parameters: *CountV, TfidfV and ClustV* are vectorizers, the first column corresponds to the number of assigned topics, the second column corresponds to the number of assigned keywords

		without tf-idf			with tf-idf		
		CountV	TfidfV	ClustV	CountV	TfidfV	ClustV
1	10	0.193	0.218	0.206	0.37	0.309	0.298
	25	0.194	0.202	0.231	0.371	0.304	0.3
	50	0.207	0.195	0.23	0.371	0.297	0.33
2	10	0.224	0.226	0.264	0.378	0.312	0.309
	25	0.241	0.22	0.202	0.376	0.301	0.316
	50	0.267	0.216	0.154	0.374	0.284	0.29
3	10	0.243	0.234	0.283	0.377	0.314	0.314
	25	0.271	0.231	0.183	0.377	0.293	0.312
	50	0.292	0.229	0.143	0.378	0.274	0.259
5	10	0.266	0.243	0.237	0.38	0.310	0.316
	25	0.301	0.242	0.157	0.381	0.284	0.28
	50	0.285	0.242	0.145	0.378	0.264	0.213
threshold	10	0.228	0.192	0.267	0.376	0.305	0.312
	25	0.250	0.184	0.234	0.381	0.298	0.319
	50	0.269	0.177	0.166	0.38	0.296	0.303

As can be seen from Table 4 we got the better results by using a combination of LDA and tf-idf keywords. Also, a built-in *CountVectorizer* from *sklearn* shows the excelling results compared with other vectorizers.

We can see from Table 4 that it is difficult to determine whether the number of

topics or the number of words affects accuracy more. However, when we use only topics with probability more than a certain threshold (a "threshold" row in the table) we obtained the same or a little bit better results compared with the other number of assigned topics. For this reason, it would be interesting to vary the threshold and investigate how it affects to accuracy.

Table 5 shows recall and precision scores for models with keywords obtained from LDA and tf-idf.

Table 5: Recall and precision of models with different parameters: *CountV, TfidfV and ClustV* are vectorizers, the first column corresponds to the number of assigned topics, the second column corresponds to the number of assigned keywords

		Recall			Precision		
		CountV	TfidfV	ClustV	CountV	TfidfV	ClustV
1	10	0.422	0.549	0.451	0.465	0.324	0.364
	25	0.422	0.572	0.445	0.463	0.327	0.374
	50	0.421	0.589	0.424	0.467	0.327	0.388
2	10	0.43	0.567	0.453	0.472	0.325	0.377
	25	0.429	0.597	0.427	0.472	0.33	0.386
	50	0.429	0.636	0.388	0.472	0.332	0.356
3	10	0.431	0.579	0.455	0.469	0.325	0.384
	25	0.429	0.626	0.432	0.474	0.335	0.403
	50	0.435	0.67	0.354	0.474	0.334	0.351
5	10	0.430	0.602	0.441	0.472	0.329	0.394
	25	0.429	0.664	0.404	0.471	0.338	0.363
	50	0.431	0.71	0.34	0.469	0.332	0.303
threshold	10	0.425	0.541	0.448	0.472	0.32	0.365
	25	0.427	0.564	0.439	0.471	0.32	0.367
	50	0.425	0.593	0.411	0.468	0.324	0.379

As can be seen from Table 5, for a model with *CountVectorizer* ("CountV") the values of recall and precision do not depend much on the number of assigned topic

and assigned words. With the increase in the number of assigned topics and words the values of recall for model with custom Tfidf Vectorizer ("TfidfV") grow, whereas for model with Cluster Vectorizer ("ClustV") the values of recall decrease. Also, we got high recall and low precision for model with custom Tfidf Vectorizer ("TfidfV") that means that the model predicts many labels and the most of them are incorrect.

As we mentioned in the section 3.3 we varied the number of topics in LDA model between eight and 50. Figure 7 shows that there are no strong oscillations in the metric values as a function of the number of topics. However, the better results are obtained for the number of topics from 20 to 40. It would be interesting to measure how the ratio of the number of topics and the number of ontology topics affects the values of metrics.

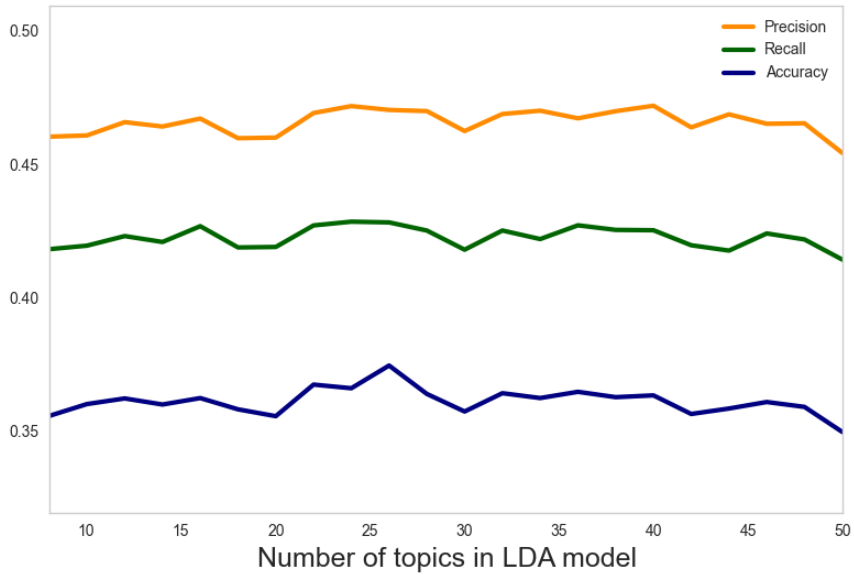

Figure 7: Accuracy, precision and recall over the number of LDA topics

4.4 Justification and visualization of results

To justify that obtained keywords are reasonable we visualized a part of one of the texts highlighting words that have a higher weight in the SVM classifier.

As an example, let us look at the abstract of an article "Next generation transcriptomes for next generation genomes using est2assembly". According to the cite *bio.tools*, its "Operation" topic is "sequence assembly". We got the same topic by using our method. Table 6 shows keywords obtained from LDA and tf-idf and Table 7 depicts top words that have a higher weight in the SVM classifier for a topic "Sequence assembly".

Table 6: Keywords obtained from LDA and tf-idf

	Keywords
LDA (the first topic)	read, assembly, sequencing, base, genome, annotate, length, map, coverage, error
LDA (the second topic)	data, database, provide, user, information, annotation, tool, available, include, link
tf-idf	assembly, est, mira, contig, assembler, platform

Table 7: The top eight words obtained from SVM classifier for topic "Sequence assembly" and their weights

word	weight
assembler	0.47
contig	0.43
assembly	0.43
annotate	0.34
genome	0.23
provide	0.21
sequencing	0.1
mira	0.07
sequence	0.05

The decreasing costs of capillary-based Sanger sequencing and next generation technologies, such as 454 pyrosequencing, have prompted an explosion of transcriptome projects in non-model species, where even shallow sequencing of transcriptomes can now be used to examine a range of research questions.
...(3 lines)
Here we present a semi-automated platform 'est2assembly' that processes raw sequence data from Sanger or 454 sequencing into a hybrid de-novo assembly, annotates it and produces GMOD compatible output, including a SeqFeature database suitable for GBrowse. Users are able to parameterize assembler variables, judge assembly quality and determine the optimal assembly for their specific needs. We used est2assembly to process Drosophila and Bicyclus public Sanger EST data and then compared them to published 454 data as well as eight new insect transcriptome collections.
... (9 lines)
est2assembly is an important tool to assist manual curation for gene models, an important resource in their own right but especially for species which are due to acquire a genome project using Next Generation sequencing.

Figure 8: Abstract of an article "Next generation transcriptomes for next generation genomes using est2assembly" with only important sentences (we discussed it in Section 3.2). Unimportant sentences are removed and replaced by the dots and the number of removed lines.

As can be seen from Figure 8, almost all words obtained from SVM classifier for topic "Sequence assembly" appear in the abstract. A word "contig" does not appear in the abstract but occurs in the rest part of the text.

It is worth noting that only the first top LDA topic and tf-idf keywords correspond to the topic "Sequency assembly" while the second topic rather indicates that this text describes some application.

4.5 Discussion and possible improvements

We achieved the best result in terms of accuracy by using built-in CountVectorizer and when a data set contains not only LDA keywords but also tf-idf keywords. The accuracy score is not high, most likely because the size of the test set is twice as high as the number of some labels. Despite this, obtained results look promising. Also, some of the labels marked with our method look reasonable. For this reason, it would be interesting to discuss them with experienced curators.

In the introduction, we mentioned that a similar work was done last year. Since we used other assumptions, for example, we simplified the ontology, then directly comparing the results is not correct. The previous work could analyze short texts as well as long whereas in our experiment we analyzed only relatively long descriptions of bioinformatics tools. However, there is a modification of LDA that has been adapted to work with short texts [52], so we can add an extra step to our algorithm where we will work with short texts as well. The main advantage of our work compared to the work done last year, is that we take into account the semantics of the text.

Beyond this work, there are several points that can improve the performance of the model and which we would like to consider in the future. For example, in LDA model, we are going to take only those keywords for each topic, the weight of which is greater than a certain threshold, and then investigate how this modification affects the accuracy of the prediction algorithm. Also, varying the values of prior of document topic distribution and prior of word topic distribution may lead to better results.

In addition to the modifications affecting the latent Dirichlet allocation part of

our method, it would be interesting to use a classifier that takes into account the hierarchical structure of the ontology.

5 Conclusions

In this thesis, we described the weak points of the manual annotation and investigated the method for the automatic annotation of bioinformatics tools. As a basis of our method, we used latent Dirichlet allocation. We described the main parameters of our method and examined which parameters lead to the best results. As a result, we received promising results. Also, we discussed possible improvements to the proposed method.

References

[1] "Wikipedia statistics in English." `https://stats.wikimedia.org/EN/TablesWikipediaEN.htm`. (Accessed 18.05.2017).

[2] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[4] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 289–296, 1999.

[5] D. M. Blei, A. Y. Ng, and M. J. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[6] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, pp. 649–657, 2015.

[7] "Tools and data services registry." `https://bio.tools/`. (Accessed 18.05.2017).

[8] V. J. Henry, A. E. Bandrowski, A.-S. Pepin, B. J. Gonzalez, and A. Desfeux, "Omictools: an informative directory for multi-omic data analysis," *Database: The Journal of Biological Databases and Curation*, vol. bau069, 2014.

[9] E. J. (under supervision Hedi Peterson), "Automatic mapping of free texts to bioinformatics ontology terms." University of Tartu, Institute of Computer Science, 2016.

[10] "Edam ontology." `http://edamontology.org/page`. (Accessed 18.05.2017).

[11] K. Tian, M. Revelle, and D. Poshyvanyk, "Using latent Dirichlet allocation for automatic categorization of software," in *Proceedings of the 2009 6th IEEE*

International Working Conference on Mining Software Repositories, pp. 163–166, 2009.

[12] Y.-H. Chen and S.-F. Li, "Using latent Dirichlet allocation to improve text classification performance of support vector machine," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1280 – 1286, 2016.

[13] R. K. V and K. Raghuveer, "Legal documents clustering using latent Dirichlet allocation," *International Journal of Applied Information Systems (IJAIS)*, vol. 2, no. 6, pp. 27–33, 2012.

[14] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal, "Bridging the lexical chasm:statistical approaches to answer finding," in *Proceedings of the International Conference Research and Development in Information Retrieval*, pp. 192–199, 2000.

[15] H. Wu, R. Luk, K. Wong, and K. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Transactions on Information Systems*, vol. 26, no. 3, 2008.

[16] J. Beel, S. Langer, and B. Gipp, "TF-IDuF: A novel term-weighting scheme for user modeling based on users' personal document collections." `https://www.gipp.com/wp-content/papercite-data/pdf/beel17.pdf`, 2017. (Accessed 18.05.2017).

[17] D. M. Blei, "Probabilistic topic models," *Communication of the ACM*, vol. 4, pp. 77– 84, 2012.

[18] "Wikipedia: Latent Dirichlet allocation." `https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation`. (Accessed 18.05.2017).

[19] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh, "On smoothing and inference for topic models," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 27–34, AUAI Press, 2009.

[20] A. Ng and K. Soo, "Topic modeling with LDA introduction." https://algobeans.com/2015/06/21/laymans-explanation-of-topic-modeling-with-lda-2/. (Accessed 18.05.2017).

[21] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, vol. 31, pp. 1–9, 2009.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, p. 1532–1543, 2014.

[23] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, 2008.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of Workshop at ICLR*, 2013.

[25] "Vector representations of words." https://www.tensorflow.org/tutorials/word2vec. (Accessed 18.05.2017).

[26] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: Deep Learning for NLP (part 1)." https://cs224d.stanford.edu/lecture_notes/notes1.pdf. (Accessed 18.05.2017).

[27] R. Mundra and R. Socher, "CS 224D: Deep Learning for NLP (part 2)." https://cs224d.stanford.edu/lecture_notes/notes2.pdf. (Accessed 18.05.2017).

[28] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer, and P. Rice, "Edam: an ontology of bioinformatics operations, types of data and identifiers, topics and formats," *Bioinformatics*, vol. 29, no. 10, pp. 1325–1332, 2013.

[29] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, "Improving LDA topic models for microblogs via tweet pooling and automatic labeling," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 889–892, 2013.

[30] N. R. Coordinators, "Database resources of the national center for biotechnology information," *Nucleic Acids Research*, vol. 41, pp. D8–D20, 2013.

[31] Willy, S. Bird, E. Loper, J. Nothman, and A. Darcet, "Natural language toolkit: Punkt sentence tokenizer." `http://www.nltk.org/_modules/nltk/tokenize/punkt.html`. (Accessed 18.05.2017).

[32] G. Varoquaux, T. Rueckstiess, J. Bergstra, J. Schlueter, N. Varoquaux, P. Prettenhofer, O. Grisel, M. Blondel, and R. Layton, "Sklearn library: K-means clustering." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/cluster/k_means_.py`. (Accessed 18.05.2017).

[33] E. Loper, S. Bird, and T. Cohn, "Natural language toolkit: Tokenizers." `http://www.nltk.org/_modules/nltk/tokenize/regexp.html`. (Accessed 18.05.2017).

[34] A. Savand, "List of common stop words in various languages in python." `https://pypi.python.org/pypi/stop-words`. (Accessed 18.05.2017).

[35] P. R. Christopher D. Manning and H. Schütze, "Introduction to information retrieval." `https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html`, 2008. (Accessed 18.05.2017).

[36] S. Bird and E. Loper, "Natural language toolkit: Wordnet stemmer interface." `http://www.nltk.org/_modules/nltk/stem/wordnet.html`. (Accessed 18.05.2017).

[37] J. Nothman, "Natural language toolkit: Collocations and association measures." `http://www.nltk.org/_modules/nltk/collocations.html`. (Accessed 18.05.2017).

[38] N. Madnani and R. Al-Rfou, "Natural language toolkit: Interface to the stanford part-of-speech and named-entity taggers." `http://www.nltk.org/_modules/nltk/tag/stanford.html`. (Accessed 18.05.2017).

[39] K. Toutanova, D. Klein, C. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of HLT-NAACL*, pp. 252–259, 2013.

[40] C.-K. Yau, "Sklearn library: Latent Dirichlet allocation." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/decomposition/online_lda.py`. (Accessed 18.05.2017).

[41] I. Titov and R. McDonald, "Modeling online reviews with multi-grain topic models," in *Proceedings of the 17th international conference on World Wide Web*, pp. 111–120, 2008.

[42] L.-W. Ku, Y.-T. Liang, and H.-H. Chen, "Opinion extraction, summarization and tracking in news and blog corpora," *AAAI spring symposium: Computational approaches to analyzing weblogs*, vol. 100107, 2006.

[43] O. Grisel, M. Blondel, R. Layton, J. Wersdörfer, and R. Sinayev, "Sklearn library: Feature extraction." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/feature_extraction/text.py`. (Accessed 18.05.2017).

[44] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, May 2010.

[45] M. Blondel and H. Alsalhi, "Sklearn library: Multiclass and multilabel classification strategies." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/multiclass.py`. (Accessed 18.05.2017).

[46] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning Journal. Springer*, vol. 85, no. 3, pp. 333–359, 2011.

[47] G. Tsoumakas and I. Katakis, "Multi-label classification: an overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[48] A. Gramfort, M. Blondel, O. Grisel, A. Mueller, J. Nothman, and H. Alsalhi, "Sklearn library: Multilabel binarizer." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/preprocessing/label.py#L633`. (Accessed 18.05.2017).

[49] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[50] A. Gramfort, G. Varoquaux, O. Grisel, and R. RV, "K-fold cross-validation." `https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/model_selection/_split.py#L347`. (Accessed 18.05.2017).

[51] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004. Proceedings* (H. Dai, R. Srikant, and C. Zhang, eds.), (Berlin, Heidelberg), pp. 22–30, Springer Berlin Heidelberg, 2004.

[52] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, "Improving LDA topic models for microblogs via tweet pooling and automatic labeling," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, (New York, NY, USA), pp. 889–892, ACM, 2013.

6 License

Non-exclusive licence to reproduce thesis and make thesis public

I, Elizaveta Yankovskaya,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis "Extraction and Clustering of App Features from App Reviews" supervised by Sven Laur

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 18.05.2017