Computer Science Research Trend Analysis

Marcel Affi, Karolina Bogacka, Paulina Pacyna,
Albert Roethel, Mateusz Wójcik

January 2021

1 Background knowledge

In order to perform data analysis on a Computer Science research paper dataset, we have divided the task into a few subtasks: first, scraping of the data, then, data preprocessing and finally data modeling and analysis.

1.1 Scraping

Web scraping, as a term, encompasses various methods used to collect data from the Internet. Usually web scraping programs simulate human Web surfing to gather specific kinds of information from different websites.[1]

In this project, our group has decided to use Beautiful Soup in order to scrape papers in their digital form from various aggregator website. This Python library is mostly used to extract data from HTML and XML files, and is commonly known to drastically decrease time spent on the task for its users.[2]

1.2 Text Cleaning And Preprocessing

1.2.1 Stop Words Removal

This technique describes a way of cleaning the data by disposing of excess, unimportant information. Stop words here are defined as most commonly used in language words, like "the", "a", "me", "is" and "all". In many Natural Language Processing tasks their removal reduces the complexity of data and allows for cleaner, much more focused analysis.[3]

1.2.2 Lemmatization

A process aimed to reduce the inflectional forms of each word into a common base. Used as an alternative to stemming, although they differ quite a bit in their internal logic. Where stemming refers to a much more quicker and imprecise process of cutting off the ends of words in the hope of producing a (mostly correct) base, lemmatization consists of a much more precise vocabulary and morphological word analysis.

Because of the accuracy difference, in our work we have decided to use lemmatization.[3]

In order to conduct lemmatization and stop words removal according to the general standards, we have decided to use methods and objects provided by the nltk toolkit[4]. This toolkit is one of the most popular, especially in education, Python platforms devoted to Natural Language Processing.

1.2.3 Bag of Words

The Bag Of Words model is simple a way of representing text data when modelling text with machine learning algorithms. It describes the occurence of words within a document.[8]

1.2.4 TF-IDF

Term Frequency - Inverse Document Frequency (commonly abbreviated to tf-idf) is a technique based on the Bag of Words (BoW) model. However, whereas Bag of Words contains only the insights about the frequency of occurence of words in a document, tf-idf acknowledges also how rarely encountered the words are in the whole corpus. [9]

The beauty of tf-idf is that it automatically removes stop words, going through each file and creating its own word frequency matrix, and finally feeding it to a tf-idf. The same technique is used in tfidfvectorizer, but tf-idf works by taking into consideration all the words, which gave us a slightly more accurate result.

1.2.5 Principal Component Analysis

Principal Component Analysis (also called PCA) is often used for linear dimensionality reduction, taking advantage of Singular Value Decomposition.[12]

1.2.6 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique used to reduce dimensionality and therefore represent high-dimensional datasets in a low-dimensional space. It allows for clearer and much more informative visualization. Instead of just maximizing the variance, like PCA, t-SNE creates a reduced feature space where similar spaces are modeled by nearby points.

1.2.7 PDF Extraction

In the end, because of the format of the dataset we have chosen for our analysis, we did not have to resort to PDF extraction. However, as a part of our project creation process, we have used PyPDF2[5] in order to analyze provided example data.

PyPDF2 is a pure Python library useful for various kinds of PDF transformation, from splitting, merging and cropping the files to metadata and text retrieval.

1.3 Data Analysis

1.4 K-means Clustering

K-means clustering is known for being one of the simplest, most popular unsupervised algorithms in machine learning. It has a simple objective - to group similar data points together and analyze underlying patterns. To do so, K-means divides the data into a fixed number (k) of different clusters, based on the value of a chosen metric computed for the cluster centroid and current point, which usually is meant to be minimized.[10]

1.4.1 Latent Dirichlet Allocation(LDA)

Latent Dirichlet Allocation is one of the most popular topic modelling techniques with a simple concept behind it. The algorithm assumes a fixed set of topics, with each topic representing a set of words. LDA's goal is to discovering a mapping of documents to topics in a way that retains most of the information about words in each document.[7]

1.4.2 Gensim

Gensim is free Python library known for its speed (it advertises itself as the fastest library for vector embeddings training, period, because of the well optimized and parallelized C routines its core algorithms use), platform independence and data streaming mechanisms, which allow users to process large corpora without worrying about RAM limitations.[6]

2 Algorithm description

2.0.1 Scraping

First stage of our process takes place in the src/scraper.py script. After initializing basic scraping configuration, the scraper begins downloading specific files. Then one should run the src/date_formatter.py script to inspect the metadata of each pdf, fetch the creationDate tag and reorganise them as seen in the above dir tree. Finally, we run src/lemmatizer.py to remove special characters, numbers, lemmatize, remove stop words for every pdf file and save it in text format.

2.0.2 Preprocessing

The techniques used for data preprocessing for various models differ slightly. The basic form of data preprocessing (found in the project as src/preprocessing.py) we use is:

```
1: procedure PREPROCESS_TXT(path, clean_data)
2:     for year ← listDirectories(path) do
3:         for month ← listDirectories(year) do
4:             for file ← listDirectories(month) do
5:                 f ← open(file, "r")
6:                 f ← filterNonLetters(f)
7:                 f ← filterSingletons(f)        ▷ As singletons we understand
   words with two or less characters
8:                 f ← filterLongWords(f)     ▷ As long words we understand
   words with more than 20 characters
9:                 f ← reduceMultipleSpaces(f)   ▷ Reduces multiple to single
   spaces
10:                f ← filterStopWords(f)     ▷ As stop words we consider 100
   least commonly encountered words in dataset
11:                w ← overwriteFile(filePath, f)
12:                w.close()
13:                f.close()
```

2.1 Clustering

In order to construct a simple model of the most common words in the dataset, we first transform the dataset into a tf-idf model and then map it into a 50D vector with PCA. In order to reduce the dimensionality even further, another transformation (this time with t-SNE) is applied. The elements of the transformed dataframe can be considered main clusters of topics.

2.2 Time Series Clustering

In order to analyze the data, it first gets transformed - like in the previous example - into a tf-idf format. Then indices of words with highest relevance

get selected. For the chosen, important words mean tf-idf scores per month are computed and later stored as a dataframe with each row being one time series.

A model we'll be using later to analyze the data is a TimeSeriesKMeans, which allows the user to divide the time series data into a few clusters.

2.3 LDA

The form of LDA used in our analysis is set up in script form as lda.py, using methods from the gensim library.

First, a dictionary and a corpus are created from all document files provided by us. The dictionary encompasses all words and their specific ids, and the corpus contains the counts of words of specific id in each separate document.

Then, those objects are used to build an LDA Multicore model. So far, our chosen hyperparameters are 3 workers, 25 topics, 10 passes and a chunksize of 2000.

```
lda = models.LdaMulticore(gensim_corpus,
                          id2word=dictionary,
                          num_topics=25,
                          workers=3,
                          chunksize=2000,
                          passes=10)

lda.save("model_multicore_25topics")
```

Listing 1: LDA model

In order to compute LDA perplexity, we use a built in function:

```
lda.log_perplexity(gensim_corpus)
```

Listing 2: LDA model

However, to compute the optimal Model Coherence, we have to build a CoherenceModel for this model, corpus and dictionary, and check its results.

In the end, we use it to compute the most dominant topic per document, count the number of documents with a specific topic over time and plot it.

3 Case studies

3.1 Clustering plot

We have reduced the dimensionality of tf-idf table (via PCA to 50D space and then using t-SNE to 2D space) and plotted it. As one can see there are some clusters that appear after applying t-SNE. The green cluster of publications contains for example publications connected to computational biology (topics related to proteins, X-ray crystallography, bioinformatics). There are strong references to Machine Learning algorithm, like SVM or Convolutional Neural Network. Those publications are the most dissimilar as they refer to the different field of knowledge and t-SNE mapping confirms that. Yellow cluster also contains a lot of biological topic that utilize ML algorithms. Those publications are mostly about protein modelling, molecular modeling, deep neural networks used for various biological predictions. Blue cluster contains publications about Machine Learning in general. Topic are related to reinforcement learning, neural networks, patterns recognition, modelling, image classification etc. Red cluster is about geological publications that also use ML. Typical topic are about GPS, solar system, climate change and geospatial data.

As one can see some clusters differ significantly in their time composition. Some of them consists of publications that are relatively new - like the blue, ML cluster or older ones like green, biology-related cluster.

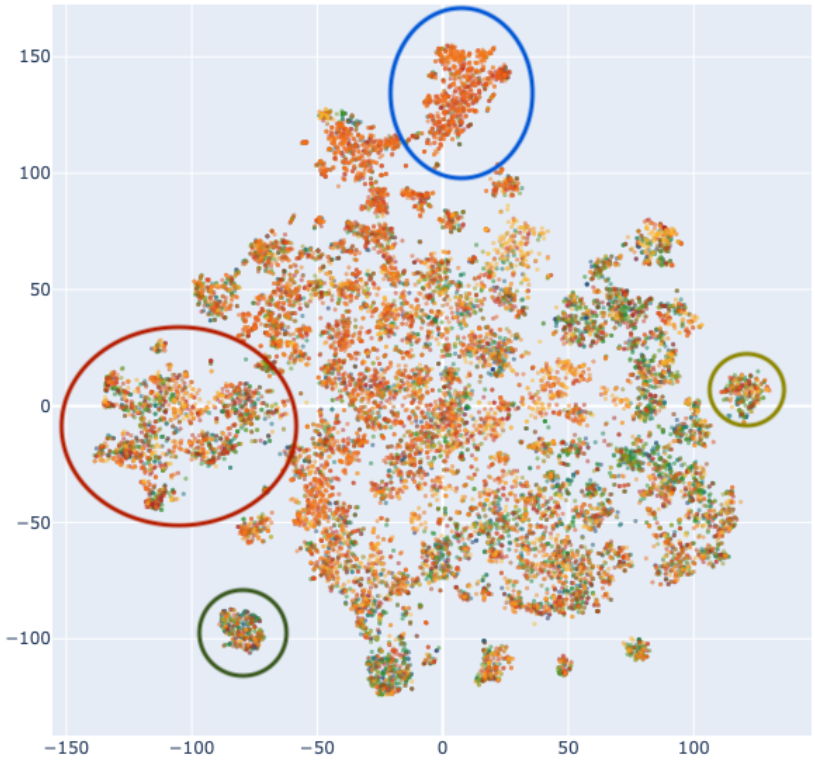


Figure 1: tf-idf clusters after t-SNE and PCA reduction. The greener publication the older, the redder the newer.

3.2 Amount of mathematics in computer science

One of the targets of our study was to answer the question "How much mathematics there was in computer science?". As an attempt to answer this question, we tried to analyse words related to mathematics in the tf-idf matrix. To obtain words related to this field of science, we can use two mathematical forums: https://mathoverflow.net/ and https://math.stackexchange.com/.
On both of these forums, each post can contain tags, which corresponds to fields of mathematics. List of all tags used on these forum, are available as an xml file on https://archive.org/download/stackexchange.

We selected 3000 most popular tags for our analysis. Examples of these tags are: *'orthonormal', 'vectors', 'foundations', 'moment-generating-functions', 'diagram-chasing', 'octave', 'lotteries', 'transcendence-theory', 'wave-equation', 'compact-operators', 'finite-groups'*. Then we computed tf-idf matrix for the whole corpus. Columns of that matrix correspond to words, and rows corresponds to each document. Below we present a subset of our tf-idf table:

	Tags		
document	function	solution	mathematics
$G^{k,l}$-constrained multi-degree ...	0.020726	0.039468	0.010714
A Case Study - Task Scheduling ...	0.020247	0.005932	0.000000

Then we selected only columns which corresponds to mathematical tags. We calculated euclidean norm for of tf-idf vector for each document. This norm is a measure of how much this document is linked to mathematics.

Document	norm
$G^{k,l}$-constrained multi-degree reduction of Bézier curves	0.089062
A Case Study - Task Scheduling Methodologies for High Speed Computing	0.006771

Each document posses a publication date. We calculated average euclidean norm of tf-idf vector per each month. We present the result as a chart below:

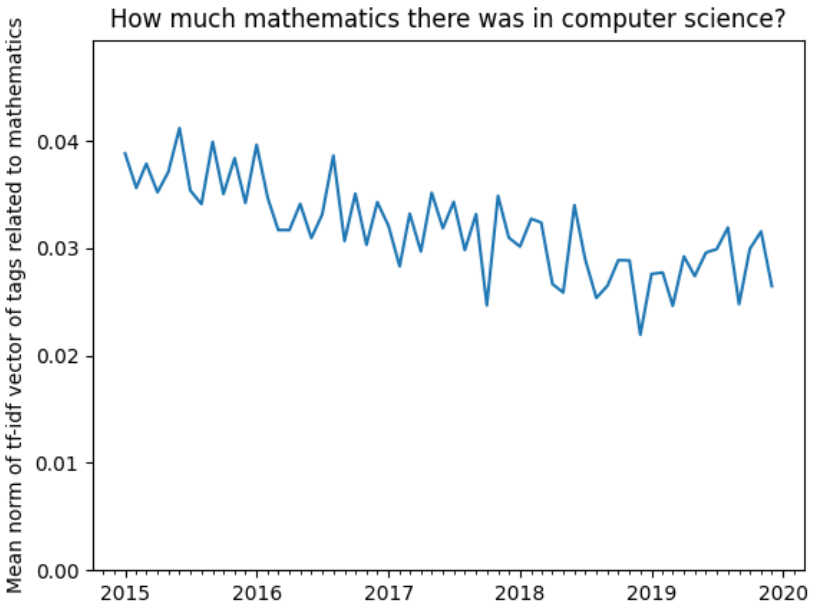


Figure 2: Average euclidean norm of tf-idf per month

3.3 Words as a time series

Time series is a series of points ordered in time. To analyse such kind of data, we can use clustering algorithms which work with time series. In our case, we can treat each word as one time series. Each columns in tf-idf matrix (which corresponds to a certain word) forms a vector of tf-idf scores. Each entry in this vector corresponds to a document, which posses a publication date. Therefore, we can compute mean tf-idf score per each month. This way, we form a time series. Below we plot time series for the most relevant words:

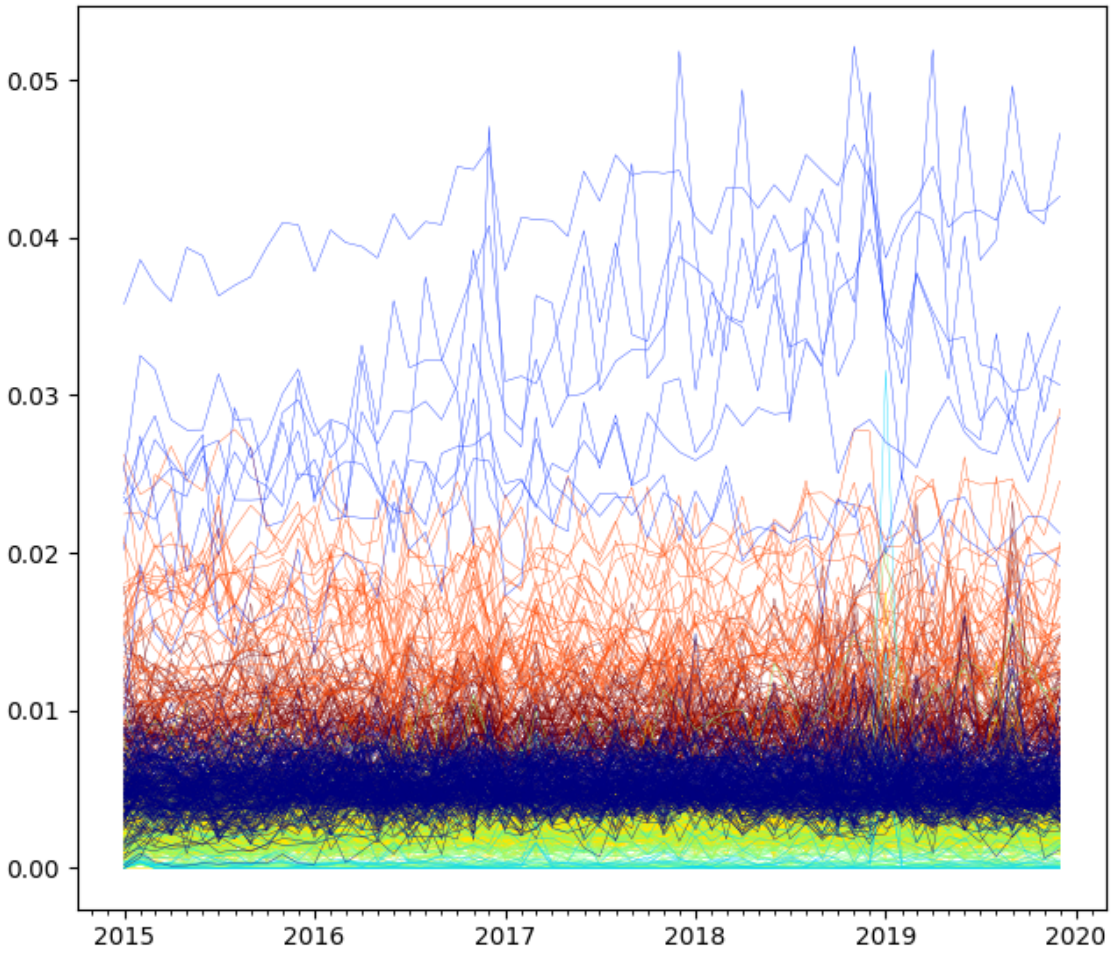


Figure 3: Mean tf-idf score per month for the most relevant words

To increase readability of this plot, we ran K-means clustering algorithm adjusted to time series data. Each cluster corresponds to words, which score behaved similarly in time. Below we present centers of these clusters and description which words belong to that cluster.

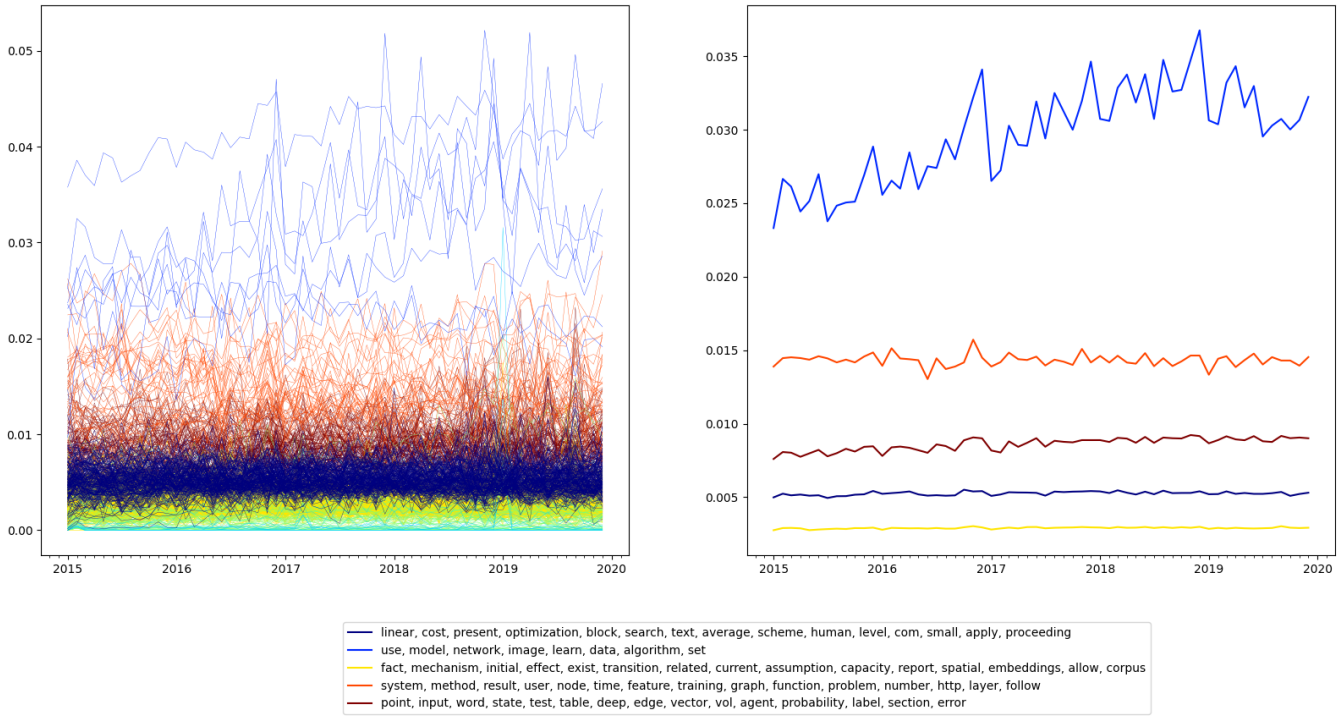


Figure 4: Cluster centers time series, labelled with words contained in cluster

3.4 Topic Analysis with LDA

First, a dictionary was created including every single word in the whole dataset (which means 1 377 000 unique ones). This data structure was used as a tactic to optimize algorithm performance and lower memory usage.

Then this dictionary has been transformed into a gensim corpus, which will later be used to build an LDA topic model, in order to divide the articles and present them in the form of topic distribution.

3.4.1 Metrics

There were two different metrics taken into account while constructing the LDA model.

First is Perplexity, a measure widely used for language model evaluation. It is measured as the normalized log-likelihood of a held-out test set, which, more intuitively, tries to approximate how well the model represents or reproduces

statistics of held out data.[11]

Secondly we focused on Topic Coherence, which is a metric that measures score of a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable and those, that are artifacts of statistical inference. Because of our processing power limitations, we have decided to focus on $C_u$mass in our analysis, which is a measure based on document coocurence counts, a one-preceding segmentation and a logarythmic conditional probability as confirmation measure.[11]

3.4.2 Finetuning

Our multicore LDA model has achieved minimum log Perplexity of around -9.01. To fine-tune the parameters, we ran the model with increasing number of topics, from 5 to 40 with a 5 topic step, and chose the number with the best coherence and perplexity values (the peak): 25.

```
model_list, coherence_values = compute_coherence_values(dictionary=
    dictionary,
    corpus = gensim_corpus,
    start=5,
    limit=40,
    step=5)
```

Listing 3: LDA model

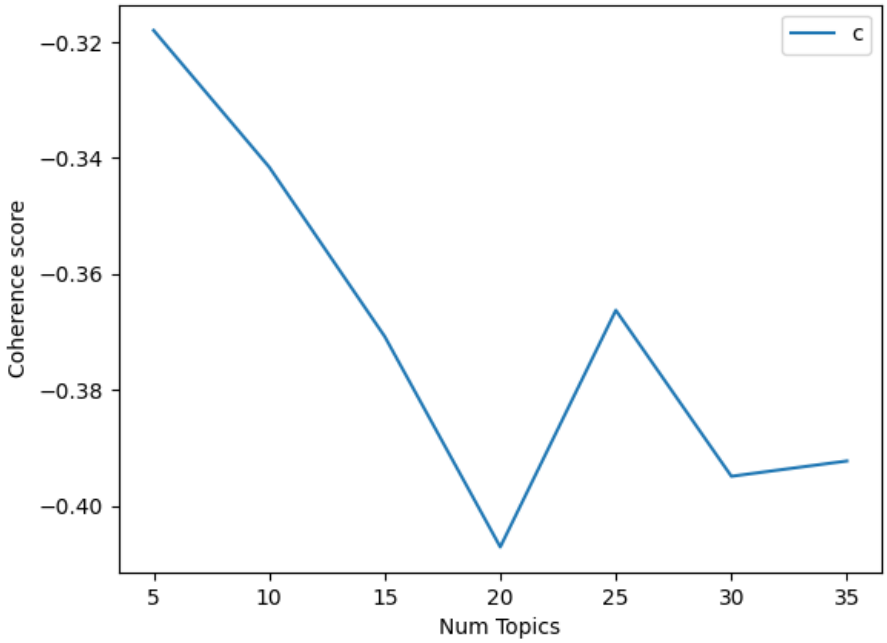


Figure 5: Coherence score per number of topics

The model coherence score per number of topics is shown on accompanying figure.

Results

Our model provided us with opportunities for versatile visualization and analysis.

Using pyLDAvis, we have constructed a handy visualization of the distance map of the topics in the form of an HTML file.

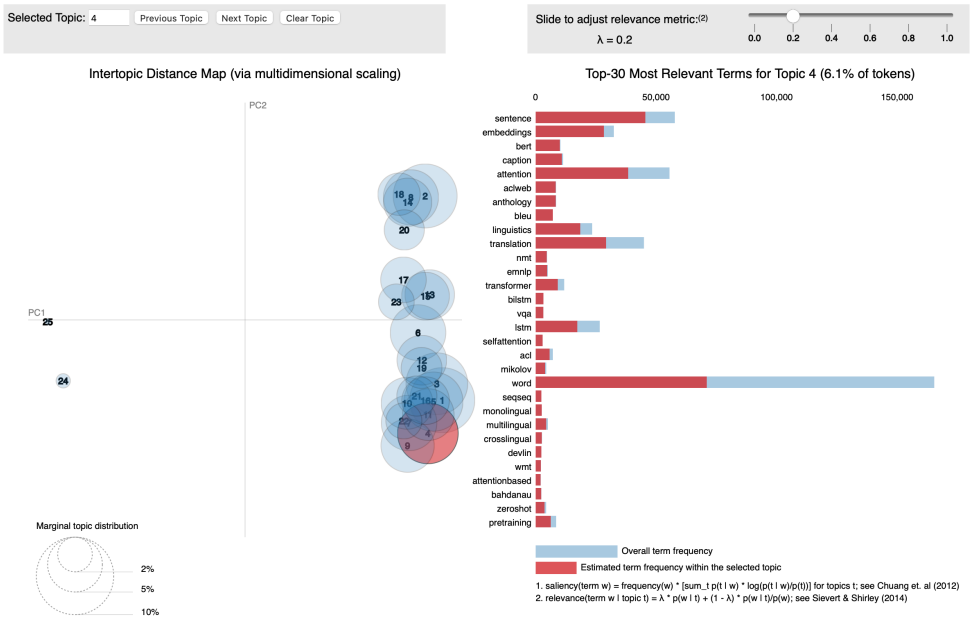


Figure 6: Coherence score per number of topics

Later, a model with the number of topics of 25 has been used to find the list of 25 most dominant topics in the dataset. Each topics is presented as a distribution of words *defining* the topic. We can present the list of topics as follows.

- (0, '0.028*"matrix" + 0.028*"code" + 0.010*"let" + 0.009*"vector" + 0.008*"linear"')
- (1, '0.018*"data" + 0.015*"use" + 0.012*"method" + 0.012*"cluster" + 0.011*"feature"'),
- (2, '0.017*"que" + 0.010*"para" + 0.009*"pour" + 0.009*"est" + 0.007*"une"'),
- (3, '0.037*"point" + 0.010*"figure" + 0.009*"distance" + 0.009*"boundary" + 0.008*"space"'),
- (4, '0.029*"algorithm" + 0.024*"problem" + 0.014*"time" + 0.013*"solution" + 0.013*"optimization"'),
- (5, '0.025*"game" + 0.017*"player" + 0.015*"strategy" + 0.014*"agent" + 0.011*"power"'),
- (6, '0.034*"data" + 0.033*"query" + 0.010*"use" + 0.009*"search" + 0.009*"privacy"'),

- (7, '0.020*"learn" + 0.014*"agent" + 0.013*"policy" + 0.012*"action" + 0.011*"task"'),
- (8, '0.016*"signal" + 0.013*"use" + 0.009*"model" + 0.009*"speech" + 0.006*"system"'),
- (9, '0.044*"user" + 0.014*"item" + 0.010*"citation" + 0.007*"video" + 0.007*"use"'),
- (10, '0.026*"network" + 0.025*"learn" + 0.020*"model" + 0.016*"training" + 0.013*"neural"'),
- (11, '0.011*"use" + 0.010*"attack" + 0.009*"http" + 0.008*"security" + 0.007*"system"'),
- (12, '0.015*"system" + 0.013*"data" + 0.009*"use" + 0.008*"network" + 0.007*"model"'),
- (13, '0.020*"system" + 0.017*"control" + 0.013*"model" + 0.012*"state" + 0.011*"time"'),
- (14, '0.053*"centrality" + 0.015*"betweenness" + 0.012*"closeness" + 0.011*"bloom" + 0.011*"function"'),
- (15, '0.013*"memory" + 0.012*"use" + 0.008*"http" + 0.007*"time" + 0.007*"performance"'),
- (16, '0.020*"channel" + 0.012*"network" + 0.010*"user" + 0.009*"ieee" + 0.009*"power"'),
- (17, '0.018*"set" + 0.013*"state" + 0.012*"let" + 0.009*"follow" + 0.009*"proof"'),
- (18, '0.050*"image" + 0.013*"method" + 0.012*"use" + 0.010*"feature" + 0.008*"result"'),
- (19, '0.017*"object" + 0.015*"use" + 0.010*"frame" + 0.010*"video" + 0.008*"pose"'),
- (20, '0.021*"model" + 0.011*"use" + 0.009*"word" + 0.007*"task" + 0.007*"language"'),
- (21, '0.016*"bound" + 0.012*"function" + 0.012*"theorem" + 0.012*"proof" + 0.011*"lemma"'),
- (22, '0.044*"graph" + 0.035*"node" + 0.025*"edge" + 0.022*"vertex" + 0.018*"algorithm"'),
- (23, '0.010*"http" + 0.010*"use" + 0.008*"word" + 0.007*"social" + 0.006*"org"'),
- (24, '0.020*"type" + 0.016*"program" + 0.013*"rule" + 0.009*"use" + 0.008*"function"')

As we can see, many of the above generated topics can be intuitively connected to existing topics in Data Science research. Topic number 22 exhibits a clear connection to graph theory and 7 to reinforcement learning. Research about neural networks can also be found in the topic number 10, while image processing in the topic number 18. To make the output of the model more clear, we have also created word clouds presenting the words defining particular topics.

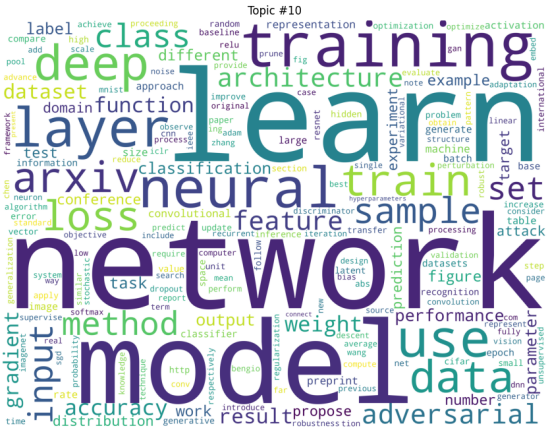


Figure 7: Word cloud concerning neural networks topic.

We can notice words connected to deep learning, training, models, neural networks and layers. In the next cloud we can observe connection with reinforcement learning.

The obtained topics have been used to plot the distribution of dominant topics in our dataset over time.

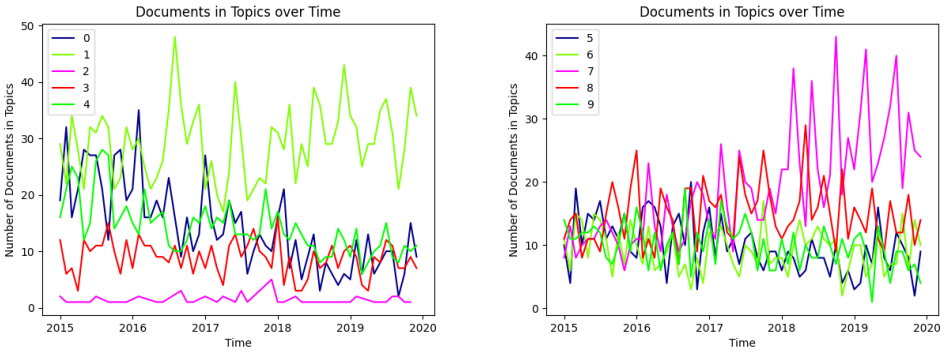


Figure 8: Distribution of topics 0-9 in time

In the presented plots, we can observe increasing popularity of already men-

tioned topic 7 connected to reinforcement learning and intelligent agents. Word cloud of this subject is presented as follows.

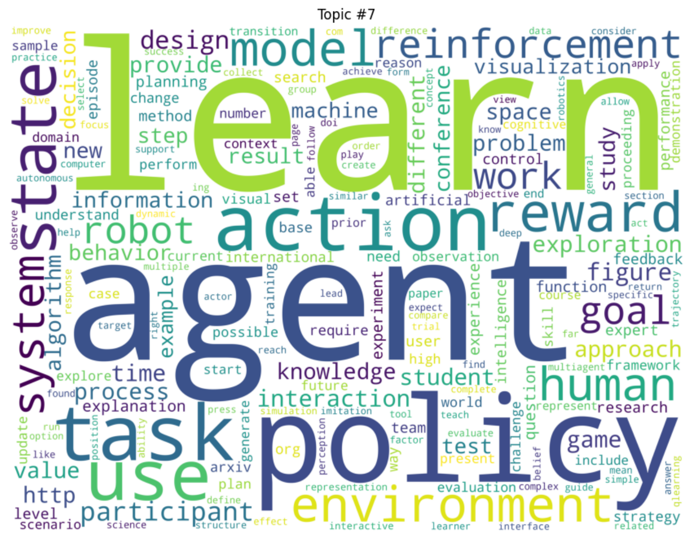


Figure 9: Word cloud concerning reinforcement learning.

Let's also have a look at the distribution of topics 10-14.

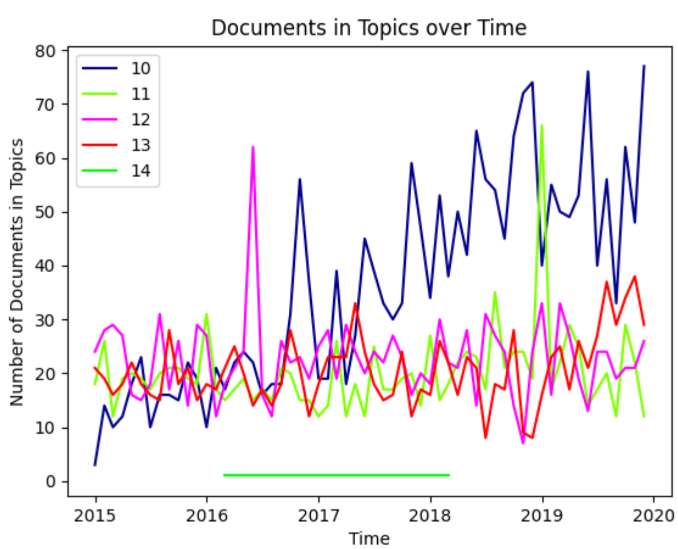


Figure 10: Word cloud concerning reinforcement learning.

We can note increasing trend of neural networks research which is a *hot* topic in data science nowadays. In the plot, we also see the outlier topic no. 14 that created a distinct topic. This one is connected to graph theory and graphs' properties, for instance centrality, betweenness, closeness. We also prepared smoother distribution counting number of topics in each year.

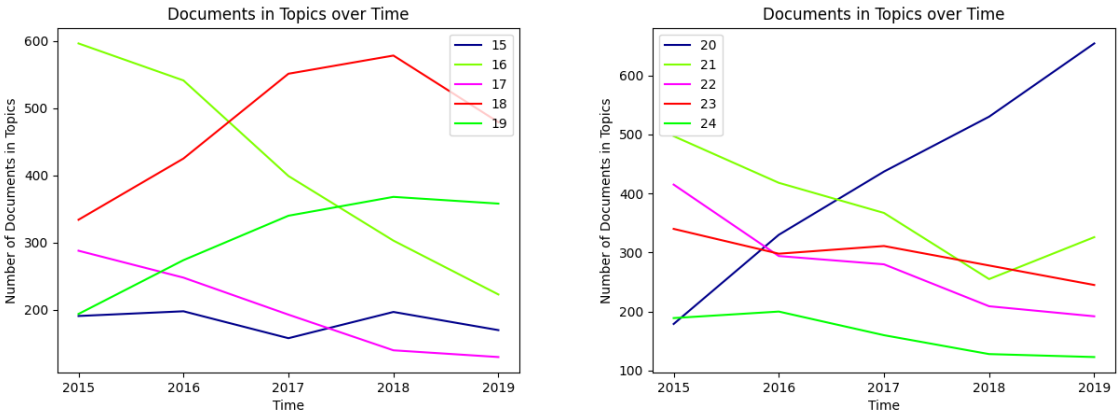


Figure 11: Distribution of topics 15-24 by years

In the plots, we clearly see the increase in popularity of topic 20 and the decrease of topic 16. The first one covers natural language processing, text and topic modelling topics. The second of which is connected to networking, signal transmission and wireless connection. Some word cloud visualizations of these topics are presented on the next page.

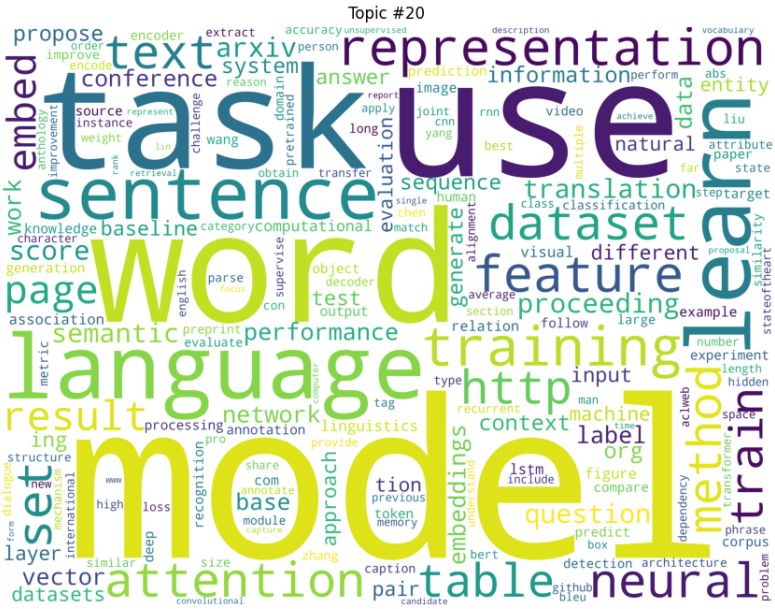


Figure 12: Language processing topic.

Figure 13: Networking topic.

It is also worth mentioning that by tweaking the following hyper-parameters of the LDA model, we were able to achieve interesting results which go hand in hand with the results of the aforementioned LDA model

- **num topics** : 10
- **passes** : 80
- **alpha** : auto
- **eval every** : 2000

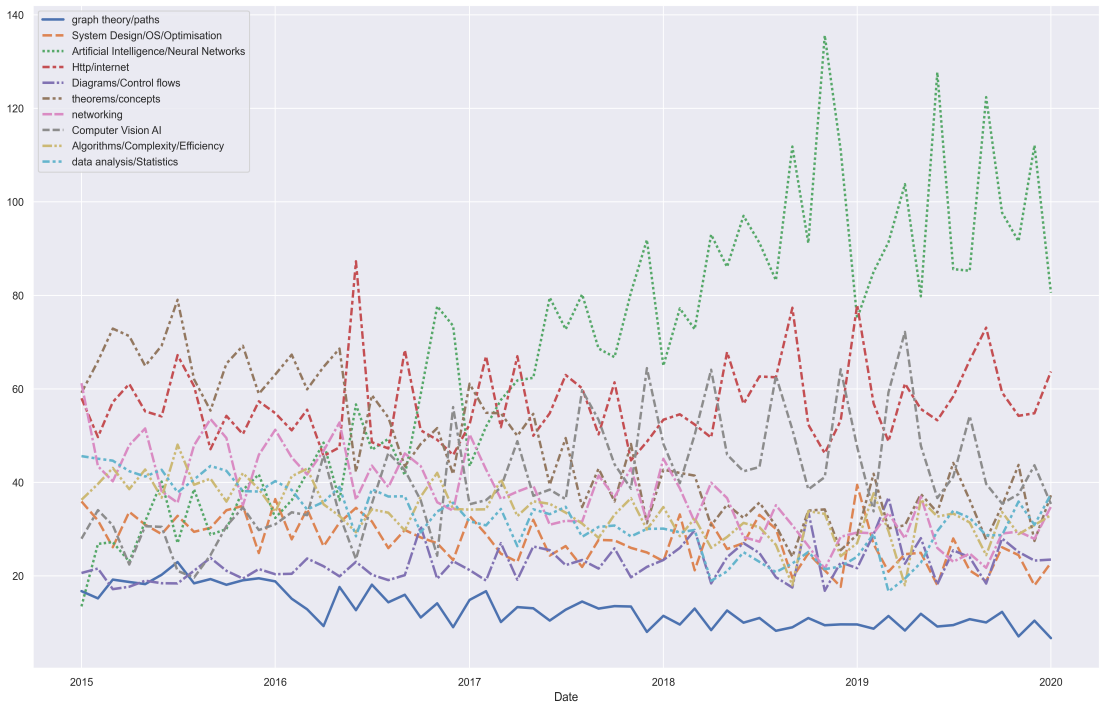


Figure 14: LDA with modified hyper-parameters (10 topics)

3.5 Testing scenarios

To scrape and preprocess your dataset place it in root directory, go to the src directory and run first scraper.py, then date_formatter.py, then lemmatizer.py and finally preprocessing.py.

To run notebooks or scripts:

- timeseries lda analysis - lda.py / lda_modified.ipynb
- clusters analysis - clusters.ipynb
- words as time series analysis - time_series_clustering.ipynb
- amount of mathematics in computer science - math_in_cs.ipynb

Our project can be found on https://github.com/PaulinaPacyna/IML-Project. Our data and models used for LDA analysis, on the other hand, are on https://drive.google.com/drive/folders/1fG-yuzZq_vhh8hk_PJw1vTZMTMAn_xjH.

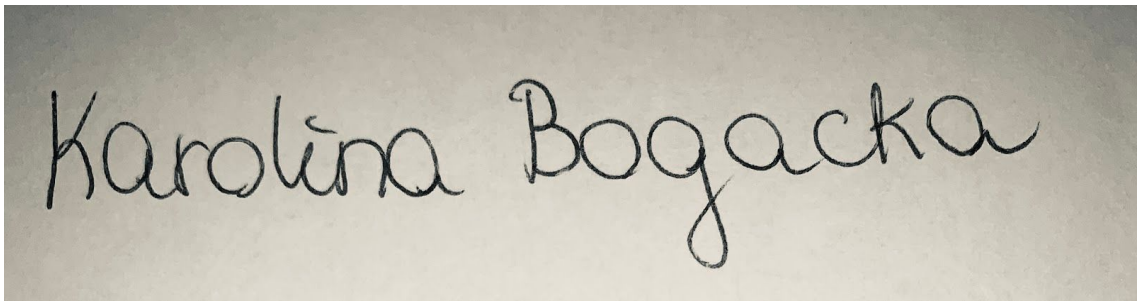
Warsaw, 02.02.2021

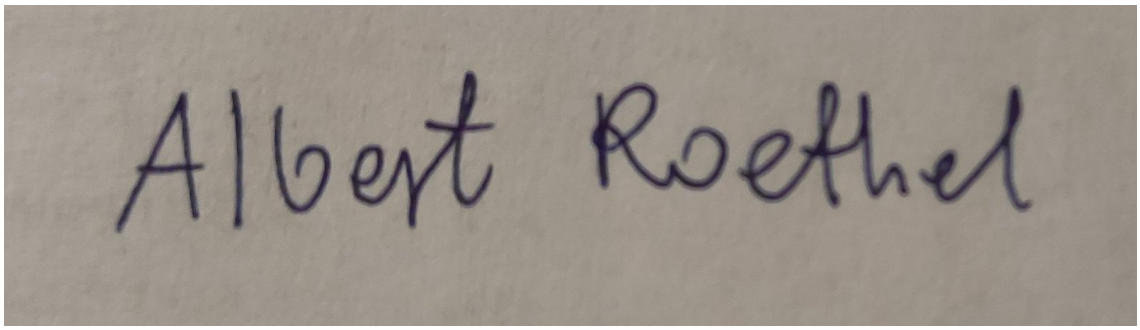
We declare that the laboratory project (*Regular project: Computer Science Research Trend Analysis*) is our own original work.

Paulina Pacyna

Mateusz Wójcik

Marcel Affi





References

[1] https://www.techopedia.com/definition/5212/web-scraping

[2] https://www.crummy.com/software/BeautifulSoup/bs4/doc/?fbclid=IwAR1qgYmcMw_zjD9Skk3Cec93xBhciyT442TJUyuDlIuw5rsvuxf3oNvsDZI

[3] https://www.crummy.com/software/BeautifulSoup/bs4/doc/?fbclid=IwAR1qgYmcMw_zjD9Skk3Cec93xBhciyT442TJUyuDlIuw5rsvuxf3oNvsDZI

[4] Natural Language Toolkit https://www.nltk.org/api/nltk.stem.html?highlight=lemmatizer&fbclid=IwAR1Wg340L0rwYD8R6P3zQpneIcfgoA-79eiRqHDvponoKhXIZd6FQ2OGnZ4

[5] https://github.com/mstamy2/PyPDF2

[6] Gensim Documentation https://radimrehurek.com/gensim/

[7] https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158

[8] https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[9] https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

[10] https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[11] https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

[12] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[13] https://miroslawmamczur.pl/jak-dziala-metoda-redukcji-wymiarow-t-sne/