Project 23Bis: Text Categorization – ACM Classification

1st Eneko Suarez
esuarez009@ikasle.ehu.eus

***Abstract*—All tasks have been completed, although, not too deeply. The results for the information retrieval system (last task) are underwhelming but its functional.**

I. Introduction

This report documents the process and findings of a project that set out to make a simple information retrieval system of scientific articles. These articles are part of the ACM Digital library. As such, some of them have already been categorized using the ACM classification.

Apart from this classification, other information about the articles is taken into account for the article searcher. This information taken into account is: author tags (or keywords), the article's abstract, its title and of course the ACM classification(s) it is part of. To see whether the information collected is useful, some prior steps are done. Moreover, the articles' are all related to COVID-19. They are all result of searching 'COVID-19' on the Digital ACM Library official website.

The scope of the project is not to make any revolutionary or close to state-of-the-art information retrieval system. Instead, the main objective is to make a simple, proof of work model using techniques learned in class. As such, not much research has been done beyond re-reading information given in the course and online searches to improve the understanding of the topics presented.

All the implementation details are available at https://github.com/esuareze/NLPProject.git.

II. Methodology

A. Task 1: scrap the data

Objective:

For every month starting from December 2019 to September 2021 (both included), retrieve title, abstract, keywords and ACM class for the first 30 results of searching "COVID-19". Store also the amount of results for every month.

Method:

To get the html for every web page used, python's standard requests library was used. Then, to get specific data from each page, BeautifulSoup library was used, which allows to make custom filters to search tags inside html. Usually the tags were searched using their class and type.

To get the data, first searches had to be made. In ACM Digital Library the searches are done with links such as: https://dl.acm.org/action/doSearch?fillQuickSearch=false&target=advancedexpand=dl&field1=AllField&text1=COVID-19&AfterMonth=12& AfterYear=2019&BeforeMonth=1&BeforeYear=2020&startPage=0&pageSize=50 . Much information is embedded in the link (such as our search term "COVID-19") but the ones we want to edit are AfterMonth, AfterYear, BeforeMonth and BeforeYear. All

of them numeric values that have to change to search for each month.

After making the search for each month (aka, requesting the html of the previous link adapted for each month), the number of results are saved and a list of the links of the first 30 results is saved.

Then, for each result, their html is requested and data is acquired using BeatifulSoup. This was specially hard due to the variety of document types, with different page structures, that the library contains. The difficulty was mostly due to the long time needed to run the program and try whether the used filters work. The long time was a result of the restrictions on web-scraping imposed by the website.

Final data structure:

The data is saved in a python list containing each article. Each article is a dictionary with the values 'monthInd': index number of the month the article was published in (0-21), 'title': the article's title as a string,'abstract': the article's abstract as a string,'authorTags': a list of all author tags (or keywords) of the articles. Each a string that can contain spaces (are not separated into words),'ccsClass': list of names of all ACM classes the articles is part of, including classes of every hierarchical level.

This data is saved in articlesData.data. It is saved using pickle, a library capable of saving and loading python objects. The decision of using this library was made to simplify data use having into account that python was the only language used in the project.

Later this data was preProcesses to fit various tasks needs.

First, stopwords (and punctuation) were removed from every string. It was saved in cleanArticlesData.data This made separating words simpler using simply a rule of separating by ' '.

Secondly, every word from cleanArticlesData.data was lemmatized and stemmed. The result of this processing was saved in preProcessedArticlesData.data

B. Task 2: Visualize amounts of results

Objective:

In the previous task, the amount of results per month was saved. This task's objective is to draw a graph with this data.

Method:

The library matplotlib allows us to draw different types of data visualizations such as graphs. It is used in conjunction with jupyter notebooks to represent the previously mentioned data. The implementation details are straight forward and not worth discussing in detail.

From this point forward, most tasks involve obtaining some data and then visualizing it (or a subset). In the project, each part of the task is separated. First the data calculation part(s) are inside python documents named after the task: taskN.py (where N is the task number). Visualizations, however, are not done in python documents but in jupyter notebooks (.ipynb) and are named taskNVis.ipynb

C. Task 3 and 4: Top 10 most used keywords

Objectives:

Task 3: Make a script to determine the 10 most common keywords among all the collected articles. On the same plot as task 2, draw the evolution of the amount of articles they are used in.

Task 4: Similarly, make a script to display the histogram of the CCS Concepts (ACM

classification keywords), and plot that shows the timely evolution of the five most common CCS concepts.

Method:

First, for each keyword, a list was made containing the month index of each article that contained such keyword. This was done by iterating through the articles and adding their monthInd to each keyword in the article's corresponding list. This lists were saved in an dictionary where the key to the lists were the corresponding keyword.

Although complex keywords (containing multiple words) were kept, all keywords were converted to all-lowercase to standardize their use to avoid multiple copies of the same keyword.

After this step, the keywords were sorted by list length to obtain the top 10 most used keywords. For these, a further step was done. For each keyword a list containing the amount of times the keyword was used in every month was made to ease its visualization. This task was easy as the data saved for each keyword was the month index of every article that used it. This data was then saved under QuickData/topKwCounts.

Finally, the visualization is once again done using matplotlib.pyplot.plot in a jupyter notebook. No further processing was needed for the visualization of the third task. Complementary visualization of amount of articles that have at least one keyword was done in "Keyword Usage Per Month.ipynb". It was calculated from the original data-set. The counts of each month were saved in a list. Each article could "count itself" in their respective month using their monthInd and checking whether their keyword list was empty or not.

For task 4, the same can be done by simply changing keywords by CCS Concept and obtaining the data for them.

In this case the visualization of the amount of articles that have a ccs class per month is included in the Task4Vis.ipynb document.

D. Task 5: title and keyword overlapping

Objective:

For each CCS Concept, determine the proportion of string matching between title and keywords. Then display it for each main CCS Concept.

Method:

The data used for this task was preProcessedArticlesData, as the information we care about is whether keywords and title share meaning, not exact word matching. In other words, we want to know whether the title and keywords refer to the same topics/ideas, so we take all words with the same step/lemma as the same.

First the same method as in task 4 was used to determine the "main CCS Concepts" (top 5 most used), except, for each concept, instead of listing the months the articles were published in, the articles themselves were saved.

Then, for each top concept, the articles that don't contain keywords are filtered out (all of them contain title so no filter is needed for it). The percentage of articles that contain keywords is saved for each concept as it is considered interesting data for future discussion.

Finally, the string matching is done both ways: how much of the title is contained on the keywords and how many keywords are contained in the title. For this, the comparison

is made at word level. This means breaking every title into its composing words and every keyword into its composing words.

The matching information is saved in various ways. First, as an average of the percentages of each article (both title words contained in the keywords and vice versa). Secondly, as a percentage of every title and all keywords (inside the corresponding CCS Concept).

Final data structure:

A dictionary where the keys are the top 5 most used CCS Concepts. The values are dictionaries with the following information:

'avgPTitleInKw' : average across articles of the percentage of words from the title that are contained in the article's keywords.
'avgPKwInTitle' : average across articles of the percentage of words from the keywords that are contained in the article's title.
'totalPTitleInKw' : percentage of words from all the articles titles that are also in their respective article's keywords.
'totalPKwInTitle' percentage of words from all the articles keywords that are also in their respective article's title.
'pArticlesWithKw'] : percentage of articles that have any keyword.

The implementation of the visualization for this task is also not very interesting as it has more to do with the specifics of the library used than any algorithm used.

E. Task 6

Objective:

Same as task 5 but enhancing the title using not only the title but also each word's synonyms.

Method:

In this case cleanArticlesData.data was used because it allows to separate the words easily but it maintains the original words (except unimportant stopwords) to be able to look them up in the selected synonym dictionary.

The used synonym dictionary was WordNet for ease of use and access. The general algorithm is the same as in Task 5 but before doing the string matching the title is augmented with the synonyms and then both title and keywords are lemmatized and stemmed (as we still only care for the words' meaning).

F. Task 7 : overlapping between abstract and keywords

Objective:

For each CCS Concept, calculate the average overlapping between abstract and keywords. The overlapping is calcualted by first calculating the tf-idf of the text's abstract and then getting the average of the sum of the tf-idf value of each keyword of the article. Then visualize the amount of overlapping for each CCS Concept.

Method:

The task was done with preProcessedArticlesData.data to keep the amount of different words on the tf-idf matrix low and also make the keywords exactly match with the ones used in the abstract. So the reasoning is close to task 5's, as both keywords and abstract words need to match as well.

The algorithm is as follows:

1) Filter out articles that don't have abstract or keywords.
2) Separate the articles depending on their ccs concepts (don't need to get only the top concepts this time).
3) For each article:
    a) fit the tf-idf model to the abstract

b) generate vector for each keyword
c) calculate sum of values of the resulting vector of each keyword
d) calculate averages between keywords

4) calculate average between articles for each ccs concept. The list containing the overlapping for each article is also saved for each concept.

The visualization for all the concepts is nothing special, they are simply ordered depending on the overlap to make visualization easier. For the top 100, the list of overlappings of all articles per concept was used and the concepts were sorted depending on the list's length and the first 100 chosen. This removes the ones with least articles, leaving the most "stable" and reliable concepts.

G. Task 8 and 9 : overlapping between abstract and keywords 2.0

Objectives:

For task 8 the objective is to calculate the overlap between abstract and keywords using Word2Vec. You will have to generate a 300 long vector for abstract and then calculate the overlapping by calculating the cosine similarity between the abstract vector and the vector generated from each keyword. The average of the values will be the overlapping for that article. Once again it will have to be displayed for each CCS Concept. Task 9 is the same but having into account both abstract and title instead of only abstract.

Method:

First of all, Word2Vec is a (type of) model to convert individual words to numerical vectors. These have to then be combined to make a single vector for a text. In my case, this combination was done by simple averaging of all vectors. The keywords were split into their word components but the abstract was processed as a whole.

Second of all, the initialization of the model. I used a model already trained by google on about 100 billion words on their data-set of news : https://code.google.com/archive/p/word2vec/.

The rest of the algorithm is similar to the one in task 7: filter articles, classify them depending on their concept. The only difference is in the actual calculation of overlappings. Instead of calculating tf-idf, etc. First the vector for the abstract is calculated and then, for each word in the keywords, the cosine coefficient of its vector and the abstract's vector is calculated. Then, as in task 7, this scores are averaged for each article and then averaged over concepts.

For task 9, for each article, a different vector is calculated for both title and abstract and are then averaged.

H. Task 10 : Interface and information retrieval system

Obejctive:

"Design a simple interface that allows the user to test the input a text query, retrieve document from ACM library and test the results of the"

Method:

First all the articles were indexed by calculating a vector for each of them. This vector was the result of a weighed average of the title's vector, the abstract's vector and the keywords' vector. The title's and abstract's vectors are the result of averaging the vector of each of their words. The keywords' vector is the result of averaging the vector for each keyword. Each keyword got its vector the same way as the title and abstract.

The weighs given were : 5 for title, 2 for abstract and $5 * log_2(num\_keywords)$ for the keywords. These values are completely arbitrary. The log is to take into account the amount of keywords. Only one keyword may not be the best way of defining most of the vector but many keywords may be enough to justify their increased weigh.

The possibility of the abstract or the keywords not being present was taken into account. Furthermore, keywords that only contained words that are not recognized by the model are not taken into account (for num_keywords either).

The resulting vector, along with the article's index in the articles' list and its ccs concepts is saved into index.data file.

Finally, when taking the user's input: first it is pre-processed by removing stopwords and numbers. Secondly, it is converted into a vector using the same method as the one used for titles and abstracts. If in this step no word in the query is found that the model recognises, the user is prompted to input a different query with english words. This step is necessary because a vector is needed in order to be able to compare the query with the articles.

Before comparing the query with the articles, it is searched for ccs concepts. If it contains any concept name, the articles are filtered to contain only the ones with that (or one of the) concept. Finally, the vector of each article that has been taken into account is compared to the query vector using cosine similarity. The results are sorted and the top 3 results are printed.

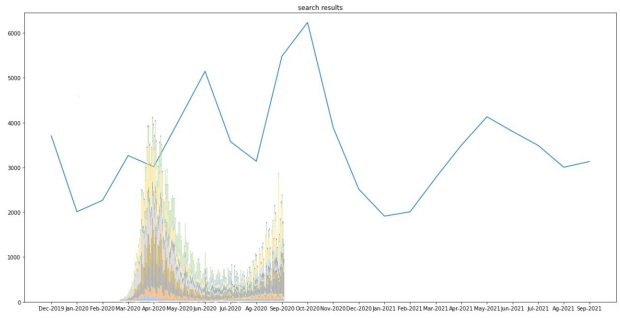

Fig. 1. Number of search results per month combined with covid death numbers

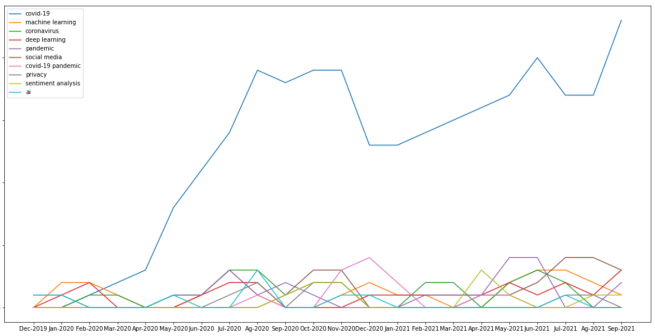

Fig. 2. Graph of top 10 most used keywords' use over time.

III. Results and Discussion

A. Tasks 1 & 2: Search Result Number

In fig 1 we can see in blue the amount of results for the search "COVID-19" per month during the months that are covered in this project (Dec 2019 - Sept 2021). If we compare them to the covid death count (overlaid on top of the graph for a subset of months) we can clearly see the pattern. Result number clearly tail the different covid waves. The explanation for new paper publishing is not only explained by public interest in the matter after each wave, these waves are also often accompanied by a new covid strains that need investigation.

B. Task 3: Most Used Keywords

In fig 2 we can see that most keywords don't appear a significant amount of times.

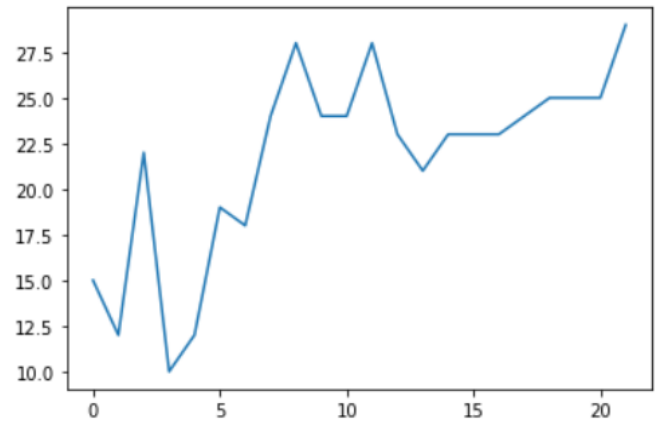

Fig. 3. Amount of articles (in our data-set) that have at least one keyword over time.

The only keyword with a significant appearance is ”covid-19”.

If we study the ”covid-19” keyword we can see that its curve resembles that of the amount of search results. The curve of ’covid-19’ usage however doesn’t fall off as much in the months between Dec 2020 and Mar 2021. This can be seen as indicative of the covid-19 staying as a topic of interest and the authors having learned to use it to grab attention, increasing the overall use of keywords as a whole (fig 3). Another hypothesis for keyword usage, apart from the amount of interest in covid-19, is the increase to attract more attention during a time of high competitiveness due to higher amounts publications.

Regarding the dominance of covid-19 over other tags, it can be explained by two factor: 1) the lack of consistency of other terms. We can see ”covid-pandemic” and ”pandemic” being used but only ”covid-19” seems to be a common tag. 2) The reliance of the search algorithm of Digital ACM Library on keywords. This makes logical sense as the clearly dominant keyword is exactly the same as our search query. This is important information when taking ACM Library’s search algorithm as an example for my own.

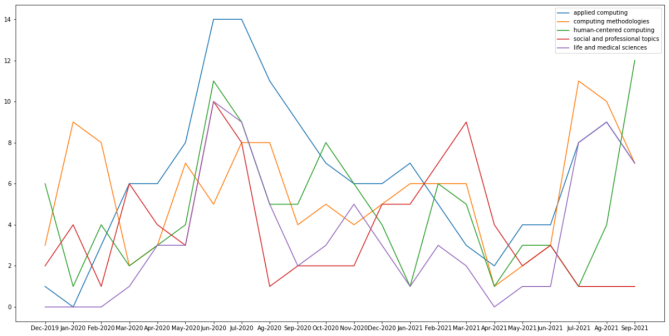

Fig. 4. Graph of top 5 most used CCS Concepts’s use over time.

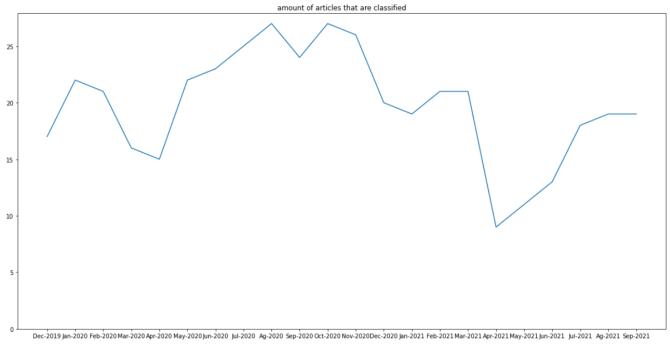

Fig. 5. Amount of articles (in our data-set) that have at least one CCS Concept over time.

C. Task 4: Most Used CCS Concepts

In fig 4 we can see the usage of top 5 most used CCS Concepts over time. Patterns in this graph are not as clear. In general they loosely follow the curve of search results but it may be pure coincidence. In fig5 we can see the amount of articles (in our data-set) that are classified. As we can see there is an increase around the first and second peak of paper publications, however it then stops following the curve, dropping off later than search results graph and then not recovering as much. This makes my previous hypothesis of authors paying more attention to paper searchability due to competition less clear.

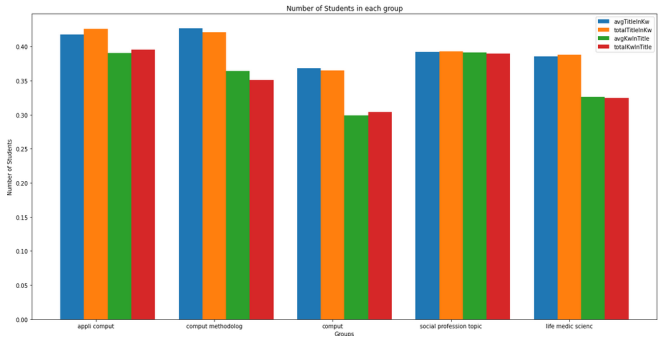

Fig. 6. Histogram of different types of overlapping between title and keywords separated per CCS Concept (only top 5 most used concepts). Blue: average over articles of percentage of words in the title that are also in keywords. Orange: percentage of words in Titles that are also in their respective keywords. Green: average over articles of percentage of keywords that are also in the title. Blue: percentage of keywords that are also in their respective article title.

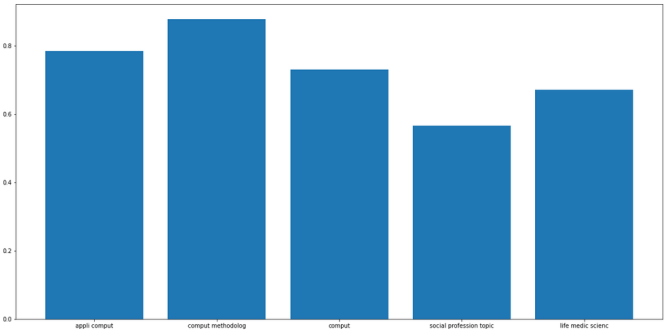

Fig. 7. for each top CCS Concept percentage of articles that use keywords

D. Task 5: Overlapping between title and keywords

In fig 6 we can see the different overlappings between titles and keywords for each top CCS Concept. The average between articles and the percentage taken as a total don't have a significant difference for either type of overlapping so we can conclude that the overlappings are close to equal between long and short titles.

Secondly, both types of overlappings (title words that are keywords and keywords that are in the title) seem to be close between concepts (within a range of $\tilde{5}\%$), so no large difference exists in title-keyword overlapping depending on CCS concept. The only difference that could be interesting enough is the reduced amount of overlap in computation concept. This concept would benefit the most from using both title and keywords.

Regarding the percentages themselves, around 40% of the title words seem to also be keywords and around 30 to 35% of keywords are also in the title. There's a remarkable overlapping between them but both of the categories seem to still be useful information for indexing as the overlappings are not close to 100%. Still, having into account their nature (being the words chosen as most import of the paper) and the percentage of their overlapping being lower, it seems keywords are more important than title. More percentage of keywords are not covered by the title than vice versa.

Lastly, in fig 7 we can see the percentage of articles that use keywords in each concept. We can see a clear discrepancy between them. Concepts more closely related to informatics (applied computing, computin methodology, computing) make more use of them than others (social proffesion topic, life medic science). So, despite keywords being a great tools for indexing, it seems we would benefiting informatics-related papers over others if we use them. We could also make the decision to use them anyway to encourage their usage in order to ultimately improve our search engine accuracy.

E. Task 6:

In fig 8 we can see the same results as in last task but with titles being augmented with synonyms. The stat of words in title that are also keywords is now worthless as it depends more on how many synonyms wordnet found in the title than actual overlapping between the two.

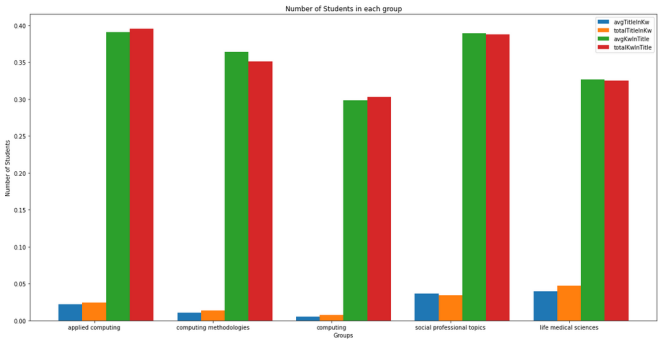

Fig. 8. Histogram of different types of overlapping between titles augmented by synonyms and keywords separated per CCS Concept (only top 5 most used concepts). Blue: average over articles of percentage of words in the title that are also in keywords. Orange: percentage of words in Titles that are also in their respective keywords. Green: average over articles of percentage of keywords that are also in the title. Blue: percentage of keywords that are also in their respective article title.

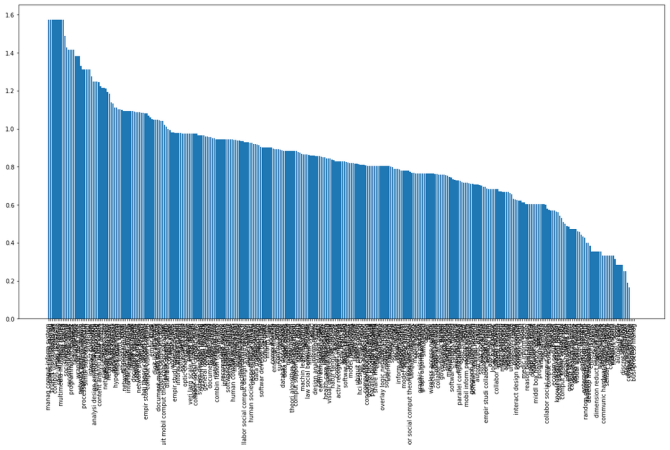

Fig. 9. overlap between abstract and keywords using tf-idf for each ccs concept

If we see the amount of keywords in the title, the numbers are pretty much indistinguishable from the ones in task 5. The overall overlapping has increased in about 5%, not a great difference. We can conclude that the wording being used in titles and in keywords is very similar and that the difference between computation and the rest of concepts its not due to computation papers using a richer vocabulary but an inherent difference in titling / keyword choosing.

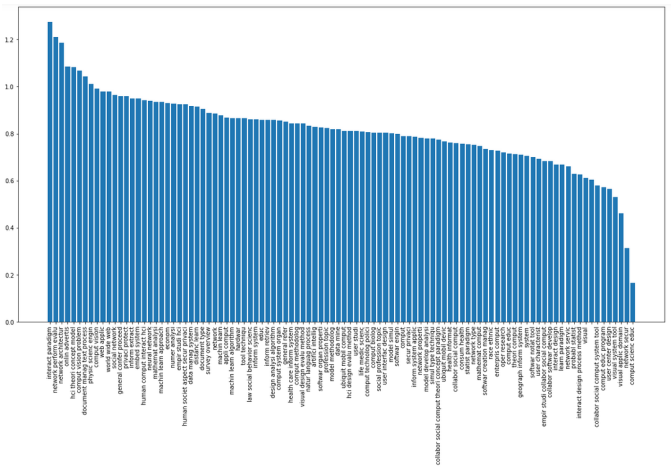

Fig. 10. overlap between abstract and keywords using tf-idf for top 100 ccs concepts with most articles

F. Task 7:

Fig 9 shows the overlapping of abstract and keywords of every CCS Concept in the data-set. Not much can be said for concepts in general as no clear pattern can be seen. Regarding overlaps in general, the variance is too much to consider it seriously, even if we only take concepts with the most articles (fig 10).

The average value of the overlapping is 0.52, which means that, on average, half of the keywords appear once. In some cases, each keyword appears more than once on the abstract. Although we could take this data as the abstract being very useful as it contains most keywords and it includes them in context, the variance on the data is too high and it wouldn't be a good way to index documents: the share of document abstracts that don't contain keywords or barely contain them is too high.

By now we have seen that not all keywords are contained in the title and neither are they in the abstract. This could lead us to think that some keywords are not related to the document and have been entered ma-

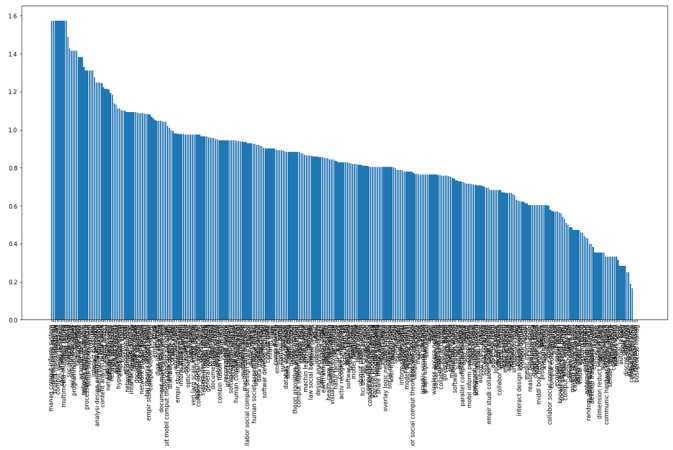

Fig. 11. overlap between abstract and keywords using word2vec for each ccs concept

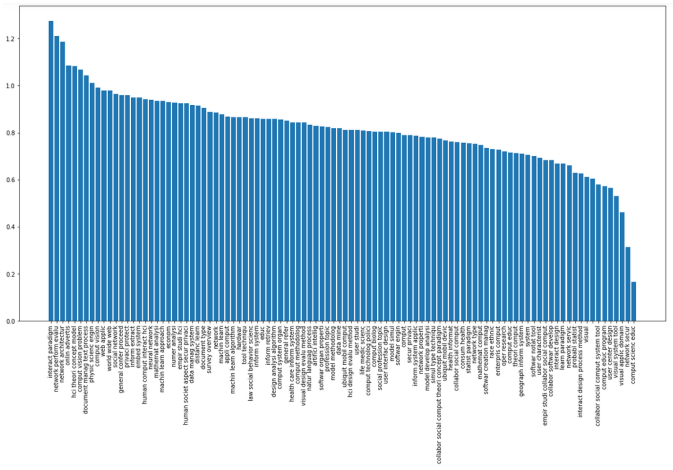

Fig. 12. overlap between abstract, title and keywords using word2vec for each concept

liciously; however, these articles have been manually reviewed before publishing so we can assume that's not the case. Instead, we can assume they're simply giving additional information from the rest of the document that the title/abstract don't contain. This is an assumption most search engines (such as google's or youtube's) can't make and should be addressed in other settings.

G. Tasks 8 & 9:

Figures 11 and 12 paint a very different picture from the one in Task 7. Not only is there less variance, but the amount of overlapping is remarkable. On average, the resulting vectors from abstract and each keyword are incredibly close. On average, the cosine coefficient of both is 0.67 if we only have into account the abstract and 0.66 if we have both abstract and title into account. The difference between tf-idf and word2vec approaches could be due to:

- word2vec is not appropriately modeling the differences between words and its classification is too broad, leading to close vectors as a result.
- tf-idf approach has too much variance and doesn't show as much overlapping because, despite preProcessing both text and keywords by eliminating stopwords, lemmatizing and stemming, we're still checking for matches of the same word (word family because of lemmatizing and stemming) so the overlap is not as consistent and doesn't have into account words with similar meanings.

Having into account the popularity of word2vec and the robustness of the word2vec model used, I would argue the second reason is more probable. As such, word2vec will be used to determine similitude between query and document texts.

Furthermore, we can see that the overlapping barely changes if we include the title.

Still, the overlapping between title and abstract, despite being relatively high, is still not high enough to faze one of them out: the abstract is proven

H. Task 10 : Interface and information retrieval system

This task was rushed and not enough though and time to experiment was given to it. The results are frankly underwhelming.

Some of the improvements that could be made if the project was to be continued:

- It would be interesting to augment the query with synonyms when searching for ccs concepts. It should yield concepts that the user is interested in more easily. The query shouldn't need to be augmented when generating the vector as the vector's position should indicate the general meaning of the query. Adding synonyms in this step would probably just give more weigh to words with more synonyms than others.
- There's no error correction or suggestions. The query is taken at face value without checking if any word could be easily fixed to another.
- The model doesn't recognize many of the words used in the corpus. Because it was trained on news, many of the technical terms or special names (such as programs names) are not recognized and thus not taken into account. This affects specially keywords and user query. Title and abstract are also affected to a lesser extent.
  Training a word2vec model is out of the scope of this project. However, if I were to do it again, I may have used a tf-idf model instead; as its easier to fit to the current corpus. This limitation was not given enough thought when thinking about the process.
- The interface is lacking. Some could say is non-existent. The console interface is minimal and barely functional. If the team had a person with a different specialization and more experience with GUIs, a better job could've been done.

IV. Overall Discussion

The data-set extracted is clearly not enough for truly significant analysis. The amount of articles per month is obviously lacking and makes statistics gathered using this data not completely reliable. Only clear patterns that are shared among all the data gathered can be relatively reliable. However, patterns not as clear like the ones in tasks 3-4 don't have much weigh : the data is further split into CCS Concepts, making the variance even higher and the confidence lower.

Moreover, for many of the tasks, a subset of the data-set is used because samples need to either have keywords, or CCS concepts or abstract or a combination of them. Something that not all documents have. For example only 71% of the documents have both abstract and keywords.

Furthermore, the timeline chosen for extraction encompasses the topic of 'COVID-19' but is insufficient for analysis of greater trends encompassing the behaviours of researchers as a whole over time. As an example, in task 3, a hypothesis is made about the overall usage of keywords, insinuating that the increase could be a consequence of researchers learning to use different tools available to raise in the search results in a competitive environment where many papers about the same topic were being published. the available data-set's timeline makes it impossible to see whether this trend is global (researchers using more keywords in general), or its simply something that happens with any hot-topic or if its a phenomenon particular to covid-19. Moreover, this hypothesis is based on such few points of data that it can't realistically hold itself against any crit-

icism.

Finally, the data is skewed. Because the documents gathered are the result of a search engine; if said engine uses keywords, ccs concepts or abstracts to evaluate similarity (which it probably does) and depending on the way it evaluates them; the data could be skewed towards documents that have such features. Making the statements and assumptions about percentage of documents that contain these features and their overall use flawed.

The decision to keep the project's specification this way is understandable when having into account the various limitations of the project in both time and resources. However, I think that improvements can be made on the project's specification regarding this issue. Although the scraping of the data was a lengthy task, for groups able and willing to invest more time into the project a greater amount of documents or a better way to gather them (such as requiring the articles to have keywords or abstracts) can be suggested.

Special attention wasn't paid to optimisation as it wasn't necessary for this scale. Upon a quick examination, in tasks 7-9, some articles were processed multiple times because they were processed in each concept they were part of. On the other hand, articles without a concept weren't processed, saving some time in contrast to a possible another implementation that first processes the overlapping of all articles to then classify them depending on concept.

V. Conclusion

The final product was rushed and not-that-well thought out. With more people in the team, different opinions could have been formed and more expertise used on the data analysed, making the overall use of the data extracted greater. Moreover, without appropriate time for such project, not much research has been done and not much experimentation has been made on the final product. With more people and time more attention to detail could have been paid in each task.

The code is not the cleanest and its not well documented but its understandable and simple. The project structure is reasonably clearly explained and is intuitive if one has the project's specification in mind. The data extracted, both raw (scraped) and processed (different metrics calculated), has been saved for future use, although in a very specific format.

Despite all of this, all tasks were accomplished satisfactorily in a reduced amount of time. The main objective of rehearsing topics discussed in class has been achieved. Valuable experience in the practical side of NLP and information retrieval systems in particular has been gained. The overall performance of the project is deemed sufficient for each task.