Large Scale Data Engineering: Scientific Plagiarism

Max Raams
m.raams@student.vu.nl
2711022

William Ford
w.a.ford@student.vu.nl
2712009

Lars de Jong
l.m.w.de.jong@student.vu.nl
2707612

1. INTRODUCTION

The Cambridge English Dictionary defines plagiarism as: "the process or practice of using another person's ideas or work and pretending that it is your own"[4]. Being one of the most serious forms of scientific misconduct, plagiarism is far from valued in the scientific community. When exposed, a scientist or journalist that has committed plagiarism will almost certainly lose their job. Because of this reason, it is important that scientific papers are checked on plagiarism so it is avoided at all costs.

The goal of this project is to build a big data processing pipeline that is able to crosscheck millions of existing scientific papers on direct plagiarism and mosaic plagiarism. Direct plagiarism is when a text is copied word-for-word from another source, while mosaic plagiarism changes the words or structure while still using the essence of a sentence from another source[3].

The data set used for this project is provided by Sci-Hub, a controversial project aiming to "remove any barrier which impeding the widest possible distribution of knowledge in human society"[5]. The Sci-Hub website provides mass access to research papers from manifold disciplines. An archive of approximately 2.4TB of scientific papers has been extracted for the big data analyses that will be executed during this project.

The remainder of this paper is structured as follows: Section 2 is devoted to a summary of previous work related to this project, discussing the algorithms and techniques used to analyse and process the data set. The goals of the project are established in Section 3. It contains questions that will hopefully be answered by the outcome of the project. In Section 4, the setup of the project is discussed. It goes over how the original data set is structured and what the resulting data set looks like. Then, the pipeline that was used to transform the original data set into the desired one is discussed and the results are showcased. Section 5 draws conclusions based on the research questions and the resulting data. Finally, Section 6 reflects upon the course of the project and discusses briefly what could be improved in a following iteration of the project.

2. RELATED WORK

To be able to analyse the data effectively and efficiently, it is important to look at some relevant work in the various fields regarding computer science. This section discusses the two most relevant scientific papers this project was based upon and how the techniques which they describe are put to use.

2.1 Smith-Waterman Algorithm

Comparing two texts in the search for possible plagiarism requires an algorithm which computes a similarity score between two strings. One of such algorithms is the Smith-Waterman algorithm[7]. This dynamic programming algorithm was first designed for use in the field of molecular biology to find similar sub-sequences between two DNA, RNA or protein sequences[6]. However not its primary intended use, it can also find similar regions between two bodies of text by comparing characters, words or paragraphs instead of nucleic acids. This is how the algorithm will be put to use in this project.

The Smith-Waterman algorithm performs a local sequence alignment on two lists of elements. It does so using an alignment matrix of size $n \times m$, where n is the number of elements in the first list and m is the number of elements in the second list. The first column and row of the matrix are initialized to zero, after which the rest of the matrix is populated with values calculated with the formula in Figure 1. Essentially, the matrix represents a context-sensitive comparison between all elements of the two lists, as is demonstrated in the example in Figure 2.

$$H_{ij} = max \begin{cases} H_{i-1,j-1} + s(a_i, b_i), \\ max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases}$$

Figure 1: The formula used to calculate scores of each cell in the Smith-Waterman alignment matrix, where $s(a, b)$ is the similarity score between elements a and b, W_k is the penalty of a gap with length k, and H is the resulting matrix.

Finally, after all values in the matrix are calculated, a back trace is executed. The trace starts at the cell with the highest value in the matrix, following the arrows representing the chosen cases backwards, and ending at a cell with value 0. The cells visited in this back trace are the optimal local alignment. The highest value in the matrix is the similarity score of the optimal alignment segment of the two input lists.

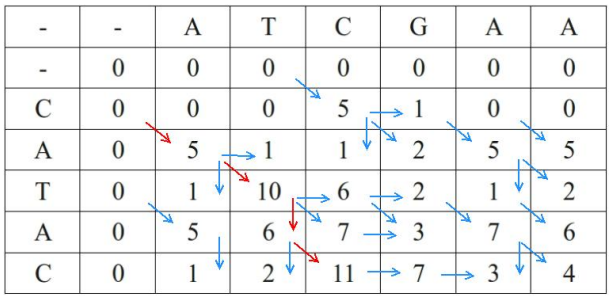


Figure 2: An example alignment matrix generated using the Smith-Waterman algorithm. Two RNA strings 'ATCGAA' and 'CATAC' are compared. The values in the matrix are calculated using the formula in Figure 1 with a similarity score of 5 and a gap penalty of -4. The blue arrows indicate the chosen case from the formula, the red arrows indicate the highest-scoring alignment path: 'AT-C'.

This local alignment algorithm seems to be the perfect fit for a plagiarism checking pipeline. It can identify a subsequence in the two input texts and quantify how similar it is, without taking into account everything else about the two texts that is not similar. However, it does not scale up very well. With an algorithmic complexity of $O(nm)$ (quadratic) it is very ill-advised to do many big comparisons using this algorithm. For a data set with millions of papers, like this project, some efficiency steps need to be made.

2.2 Latent Dirichlet Allocation

The next goal is to find a way to minimise the number of comparisons between papers that need to be done, without sacrificing too much accuracy on the plagiarism detection. Latent Dirichlet Allocation (LDA) offers a solution to this problem.

LDA is an iterative topic modeling algorithm that performs an unsupervised classification with a variable number of topics on a corpus of text documents[2]. It is similar to the k-means algorithm, but LDA can assign documents to multiple classes, making it more realistic for this use case than k-means, where the sets of documents are disjoint.

Latent Dirichlet Allocation approaches a document as a collection of words, in which the order or any other grammatical structure in the text is not taken into account. Using this 'bag of words' approach, LDA calculates in iterations which words are likely to appear together in a document; it forms 'topics'. These topics can be described as a set of terms and the probabilities that a document d should be in that topic, given that the given term is present in d. As an example: it's safe to say a document containing the words 'cell', 'protein', 'gene', and 'human' has a high probability of being a research paper in the field of molecular biology, while a document containing the words 'voltage', 'circuit', 'power', and 'device' has a high probability of being a research paper in the field of electrical engineering. These were two real examples of topics that LDA formed on the data set for this project.

LDA can do this for any predefined number of topics k and does so without prior set up of the topics, unsupervised. By using LDA, it is possible to add a column to the data set containing the topics a document belongs to. This column can be used as a condition while forming the cross join: only cross-check documents that have at least one matching topic. This approach will probably not sacrifice too much accuracy on the plagiarism check, because it follows the notion of it being futile to check papers that are fundamentally dissimilar. For example, it is highly unlikely that papers on medical research will have plagiarized from papers in the computer science field. Secondly, this approach will greatly improve the performance of the pipeline. Instead of having to crosscheck all n documents in the data set, which would result in n^2 comparisons, only documents within topic cluster have to be compared. This results in a 'sum-of-squares' number of comparisons, which, in practice, will always be smaller than the total square number of comparisons. The more topics are used, the smaller the number of documents per topic becomes and thus, the smaller the total number of comparisons becomes. Using too many topics however, might lead to LDA losing accuracy and finding patterns where there are none. Experimentation with the value of k is crucial here.

3. RESEARCH QUESTIONS

The goal of this project is to create a pipeline that is able to detect plagiarism within a huge data set of research papers. Hence, the main research question will be: **"What is a proper pipeline to check for plagiarism among a large number of papers?"**. The following hypothesis can be formulated for this research question: plagiarism in the scientific community is considered a serious violation of academic integrity. Therefore, it is not expected that scientists will commit plagiarism very often, and when they do, they will probably be careful. It is not expected that there will be many occurrences of large text sections overlapping between two papers without a source cited.

Using the data produced with this pipeline, further research can be done. Some of the main research questions to be answered with this data will be:

- Which kinds of overlaps between research papers can be found?
- How scalable is the pipeline that was used to generate the data? Were the proper algorithms / tools used?
- Has the amount of overlap/plagiarism in scientific research papers changed over the years?
- What are the largest parts of overlapping text in the data set?
- Are there any authors that are repeatedly reported for overlapping text?

The first question will be answered by analyzing the various overlaps the pipeline will find, and discussing patterns that might be found within those overlaps.

For the second question, an evaluation of the pipeline is necessary. This evaluation should take into account the weak spots of the pipeline and if/how those weak spots can be fixed in a following iteration of the project.

The third question can be answered using a simple line graph on the final data set, comparing the discovered overlaps with the year that they were published. The hypothesis: the amount of plagiarism is expected to grow over the

years, especially since the world wide web started to become popular.

The fourth question will be answered using a table, showing the papers which have the top ten largest overlap of the data set. It will also be possible to review the overlapping text.

The final question will be answered using a bar plot which will show the names of the authors and the amount of overlapping text.

4. PROJECT SETUP

4.1 Input Data

The data set consists of 3.938.483 papers equally divided over 4.761 zip archives. The papers are all PDF-formatted and may contain from 1 to over 100 pages. There are a few papers in the data set with an extremely large number of pages, for example: a book with over 300 pages. An overview of the page count distribution can be seen in Figure 3.

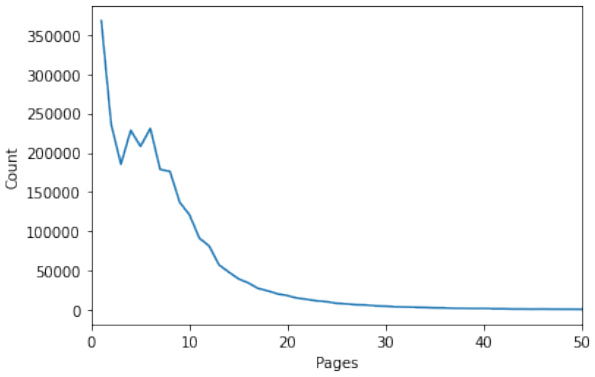


Figure 3: Page count distribution over the full data set. Papers above 50 pages are left out of the graph to show more detail in the 0 to 20 pages range.

Some papers seem to be duplicates of other papers. They have different file names, but their content is exactly the same. Some of the papers in the data set also seemed to be corrupted, as it was not possible to open them. Many papers also contain graphs, tables and images. Some papers even consist of only images. Because this project will only focus on text mining, plagiarism of graphs and other images is left out of scope.

Before the papers can be processed, they first need to be converted from PDF files to plain text data. To do this, an optical character recognition program (OCR) will be used. This will be addressed further in Section 4.3.1.

For some of the papers, important information can be found in the metadata of the PDF file. Sometimes, it contains a list of authors, the title and the publish date. This is not the case for all papers though, making it very difficult to reliably use without discarding a large part of the data set.

Instead of using the metadata for information, another option is to use Digital Object Identifiers (DOI) which are included in the filename of the papers. These are unique identifiers which can be used to look up more information about the paper in a third party database. Though there are still papers for which some information cannot be found, this is a much more reliable method compared to using the metadata. The information in this database includes the title, the author(s), the publish date and much more[8]. For this project only the following fields will be used:

- Title: string that contains the title of the paper
- Authors: list of objects that contain the given name and family name of the authors of the paper
- Publish date: string that contains the date the paper was published, formatted following the ISO 8601 guidelines [9]
- Type: a string describing the type of the paper (e.g. 'journal-article', 'book-chapter' or 'proceedings-article')
- Publisher: string that contains the name of the institution that published the paper
- Journal: string that contains the name of the journal that the paper was published in

All of these data fields will be joined with the initial extracted data and put into parquet files, an efficient data storage format, so it later can easily be read into a Spark data frame.[1]

4.2 Output Data

The output data for this project will consist of three spark data frames, stored in parquet files. These three data frames will contain the following information:

At the start of the data analysis phase (as will be discussed in section 4.3.3), each paper will be categorized into different topics using Latent Dirichlet Allocation (see: Section 2.2). Each of these topics will have a unique ID and a list of words that are related to that topic. Each word also has a probability score that indicates the probability that a document belongs to the topic, given that it contains that word. This "Topics" table is one of the output data tables.

The second data product is the cleaned data frame representation of the input PDF data. This data frame will contain all the necessary information about all the papers the pipeline could gather such information about: the DOI, title, page count, publish date, language, document type, publisher, journal, authors, cleaned text (as described in Section 4.3.2) and the topics, which is an array of foreign keys for the first data product.

The third and final data product is the data frame containing the detected overlaps between the compared papers. This data frame contains the following fields:

- DOI1: the DOI of the first paper in the comparison.
- DOI2: the DOI of the paper that the first paper is being compared with. Both DOI fields function as a foreign key for the second data product.
- Score: the Smith-Waterman similarity score of the comparison between the two papers.
- Overlap1: the overlap between the two papers from the perspective of paper 1.
- Overlap2: the overlap between the two papers from the perspective of paper 2.

4.3 Pipeline

4.3.1 Phase 1: Data Extraction

As mentioned before, the initial data set consists of many PDF files, stored in zip archives. The goal of the first phase is to extract the data from these files so that it can be processed more easily throughout the rest of the pipeline.

First of all, a list of all file names is created and put into a data frame, together with the number of the zip archive they are a part of. These numbers are later used to partition the extraction of the PDF files. Then, more columns are added to the data frame by joining it with a data frame containing more information on every DOI. Papers which are not present in the DOI data set are dropped from the data set.

Using the *PyPDF2* package for Python, all PDF files are then converted to a plain text. This package is not as accurate as some other packages, but it allows for extraction from the uncompressed zip archives without having to unzip the archive. This is important for such a large data set, as it speeds up this phase significantly. Some other packages were able to retrieve much more accurate results, but also took more time to process a single paper. During this step, the PDF reader also extracts the page count of the papers. Papers with a large number of pages will take a lot of computation time in later phases, therefore the papers with over 100 pages are thrown away. As can be seen in Figure 3 this will only be a very small percentage of the entire data set. If one of these steps fails for a particular paper, for instance, an error that occurs during the text extraction, the paper is discarded from the data set.

After extracting the bodies of the papers and putting them into a data frame, one extra column is added containing the language that the paper was written in. This language is determined using the *langdetect* Python package which checks the body of the given paper. As can be seen in Figure 4, nearly 85% of the papers in the data set are written in English, followed by German with approximately 11%, French with a bit over 1% and 4% for all other possible languages, including "could not detect". In a following step, LDA will categorize papers in topics using the words in their bodies. Therefore, it would be best to separate languages from each other. Since German and the other languages are just a small fraction of the data set compared to the English papers, they are thrown away in this phase.

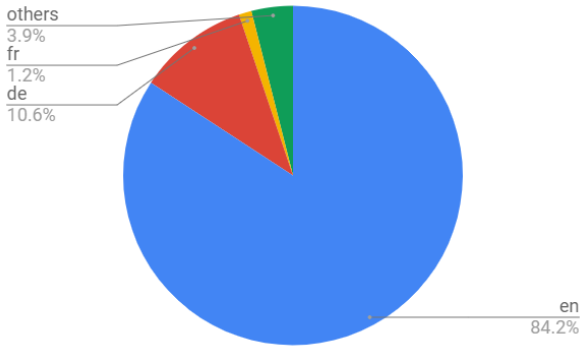


Figure 4: Language distribution over the full data set.

4.3.2 Phase 2: Data Cleaning

Now that all of the approximately 4 million scientific papers have been extracted into a single data frame, it is necessary to compress and clean the data for better algorithm performance on both time and accuracy. First of all, duplicate bodies are removed. As mentioned in Section 4.1, some papers have identical bodies, even though they have different filenames. These are mistakes on the data supplier's end and need to be filtered out to prevent false positives. After that, each paper goes through a list of steps:

First of all, every character in the body of a paper is converted to lowercase. This is done to increase the recall of the algorithm in phase 3; capitalization is not important to look for with plagiarism. Also, all punctuation and excessive white space is removed since those serve no purpose for the algorithm either.

After that, stop words are removed from the corpus. These are words that appear frequently, but provide little information, such as "I, me, we, our, am, could, most, other, very, so". Removing these words will considerably reduce the size of the data set and increase the accuracy of the algorithm in the next phase, since the data that remains will be more descriptive of the actual content of the paper. The "Natural Language Tool Kit" (NLTK) Python package contains a ready-made list of some of the most common stop words. It also contains functions to filter a corpus on that list efficiently. It is possible to extend NLTK's built-in list of stop words to create a more domain-specific list to filter on. To do this, a new list is created with the words from all the available documents in the data set after phase one, with their document frequencies. The document frequency is a metric that defines the number of documents in a data set that contain a certain term. At the top of this list will be words that appear in many different papers, and thus provide little specific context to the individual papers. It would be futile to check them for plagiarism. This new list is filtered to only contain words above a certain document frequency threshold. The list now contains words such as: "however, result, study, research, thus, also, condition". Then, the list is appended to the stop words list.

Finally, all verbs and nouns of the text body are stemmed. Stemming is the action of reducing a word to its root form or "stem". Plural nouns become singular, conjugated verbs become unconjugated. This step can also be achieved using the functions provided by the "Natural Language Tool Kit" Python package. After this step, the phase 3 algorithm should become much better at finding mosaic plagiarism, because it can focus more on the meaning and function of the word in a sentence, instead of looking at how it is spelled exactly. After all, the words "colors", "coloring", "colored" and "color" should not be differentiated between for this use case. The results of this phase will put into a new data frame and stored in a new parquet file, so that it does not disturb the raw output from phase 1 in case the phase 2 cleaning algorithm needs to be tweaked after the fact.

4.3.3 Phase 3: Data Analysis

In this final phase, the actual plagiarism detection can finally start. But before doing so, the data will first need to be be clustered using topic analysis.

The topic analysis is done using the LDA algorithm discussed in Section 2.2. Every paper will be assigned a maximum of three topics. For all papers, the probability that

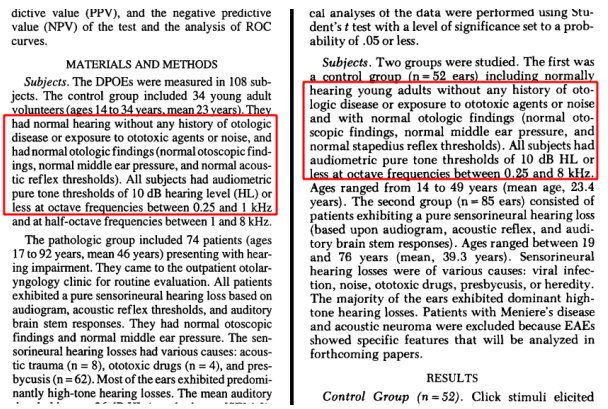
the paper is part of a topic is calculated for every topic. By setting a threshold of 0,3 on that probability, a paper can only be part of a maximum of three topics. Some papers will not be assigned any topics, as they might not contain any words that give them a high enough probability to cross the threshold of being part of any topic. These papers likely do not contain any valuable information, and thus can be discarded from the data set.

After the data has been divided into topics, the papers can be compared for plagiarism using the Smith-Waterman algorithm, as discussed in Section 2.1. For every topic, all papers will be compared with each other. To speed up this process, papers are only compared with newer papers, so that each combination of papers is only checked once. This also makes sure that the newest paper of the two is considered the one of which the author plagiarised.

4.3.4 Results

In total, 35930 sections of overlapping text were found. A good example of overlapping text found by the pipeline can be seen in Figure 5. More examples can be found on the visualisation website.

The difference in amounts of plagiarism per year can be seen in Figure 6. There are also authors that repeatedly have overlapping text with older papers. The ten authors that have the most sections of overlapping text can be found in Figure 7.


dictive value (PPV), and the negative predictive value (NPV) of the test and the analysis of ROC curves.

MATERIALS AND METHODS

Subjects. The DPOEs were measured in 108 subjects. The control group included 34 young adult volunteers (ages 14 to 34 years, mean 23 years). They had normal hearing without any history of otologic disease or exposure to ototoxic agents or noise, and had normal otologic findings (normal otoscopic findings, normal middle ear pressure, and normal acoustic reflex thresholds). All subjects had audiometric pure tone thresholds of 10 dB hearing level (HL) or less at octave frequencies between 0.25 and 1 kHz and at half-octave frequencies between 1 and 8 kHz.

The pathologic group included 74 patients (ages 17 to 92 years, mean 46 years) presenting with hearing impairment. They came to the outpatient otolaryngology clinic for routine evaluation. All patients exhibited a pure sensorineural hearing loss based on audiogram, acoustic reflex thresholds, and auditory brain stem responses. They had normal otoscopic findings and normal middle ear pressure. The sensorineural hearing losses had various causes: acoustic trauma (n = 8), ototoxic drugs (n = 4), and presbycusis (n = 62). Most of the ears exhibited predominantly high-tone hearing losses. The mean auditory

cal analyses of the data were performed using Student's *t* test with a level of significance set to a probability of .05 or less.

Subjects. Two groups were studied. The first was a control group (n = 52 ears) including normally hearing young adults without any history of otologic disease or exposure to ototoxic agents or noise and with normal otologic findings (normal otoscopic findings, normal middle ear pressure, and normal stapedius reflex thresholds). All subjects had audiometric pure tone thresholds of 10 dB HL or less at octave frequencies between 0.25 and 8 kHz. Ages ranged from 14 to 49 years (mean age, 23.4 years). The second group (n = 85 ears) consisted of patients exhibiting a pure sensorineural hearing loss (based upon audiogram, acoustic reflex, and auditory brain stem responses). Ages ranged between 19 and 76 years (mean, 39.3 years). Sensorineural hearing losses were of various causes: viral infection, noise, ototoxic drugs, presbycusis, or heredity. The majority of the ears exhibited dominant high-tone hearing losses. Patients with Meniere's disease and acoustic neuroma were excluded because EAEs showed specific features that will be analyzed in forthcoming papers.

RESULTS

Control Group (n = 52). Click stimuli elicited

Figure 5: Example of overlapping text found by the described pipeline. The text section on the left comes from "P. Bonfils, P. Avan, P. Landais, M. Erminy and B. Biacabe. Statistical evaluation of hearing screening by distortion product otoacoustic emissions", the text section on the right comes from "P. Bonfils and A. Uziel. Clinical applications of evoked acoustic emissions: results in normally hearing and hearing-impaired subjects". These papers have an author in common, which might explain the overlap.

4.3.5 Visualisation

Because it is difficult to think of exciting ways to visualise overlapping text, the visualisation website will visualise more than just the results. It guides the user through the entire pipeline used to get to the end result, visualising every step along the way with understandable examples.

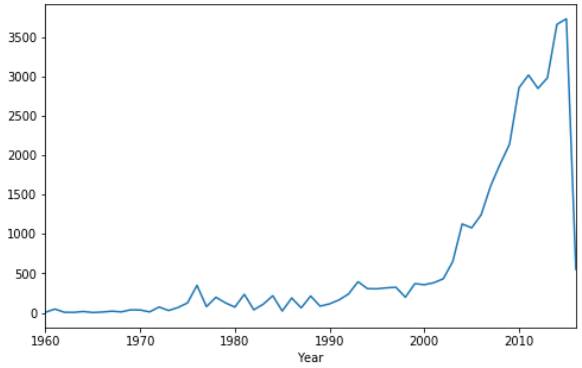


Figure 6: Graph showing the amount of overlapping text found over the years. The amount has clearly skyrocketed over the past two decennia.

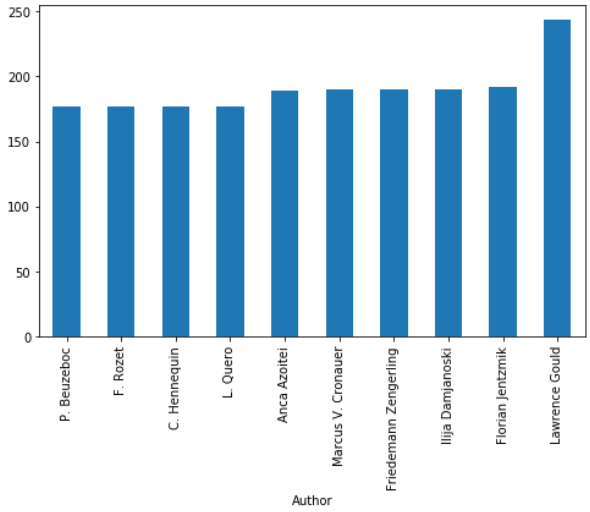


Figure 7: Graph showing the authors with the most overlapping text sections found in older papers. Note that overlapping text does not necessarily imply plagiarism.

The goal of the visualisation website is to provide access to semi-advanced information in the field of big data to people who are interested in it, but do not have the background knowledge in computer science to understand complex papers on the matter. The challenge is to format and describe the matter in layman's terms, while also not insulting the intelligence of the reader by over-explaining certain concepts.

The lay-out of the website is as follows:

- The home page provides an introduction of the subject matter to the user and shows the pipeline phases as boxes to click on.
- The cleaning page shows what exactly happens to the text in the cleaning phase. It goes over both global cleaning, and local cleaning. The local cleaning section follows a running example of how a small text section would be cleaned by the pipeline, from stripping white space to word stemming.

- The LDA page describes how the LDA algorithm creates clusters based on topics, what these topics look like and how they are used in the pipeline to improve efficiency. The page also contains a colorful bubble chart, showing a few examples of topics from the pipeline, and the five words most relevant to those topics.

- The Smith-Waterman page visualises how the algorithm is able to find similar text sections between two papers. It does this by showing, step-by-step, how the alignment matrix is filled up with values and how those values are back tracked to find text similarity, similar to Figure 2.

- Finally, the results page shows a set of examples of the actual results of the project. The results are categorized into different tables. Clicking on the DOI of a paper opens a new tab on Sci-Hub where the user can read the paper itself. Clicking somewhere else on a row in a table shows the overlap that the pipeline managed to find.

The website went through a few iterations before ending up the way it has ended up. This was done with the help of a small testing group of the target audience: people who do not have any prior knowledge in big data but are interested in learning. The feedback was directly used to gain insight in the language that should or should not have been used, certain topics that needed more explanation and the topics that should have been left out due to the high level of difficulty (e.g. the method with which LDA actually generates the topics).

The website can be found in the Databricks File System at *dbfs:/mnt/group21/website.tar.gz*, or online on *https://max-r.nl/LSDE*.

5. CONCLUSIONS

Though the created pipeline is able to detect overlapping text sequences accurately, it is not scalable enough for bigger data sets. The main problem seems to be the limited number of topics the chosen LDA implementation could handle without crashing. This failed to lower the number of comparisons to be made to a feasible value.

The overlap that was found by the pipeline can be divided into different categories, which can be grouped in two major groups: false positives and true positives.

There are many types of false positives that were found, the most frequently occurring type being overlap in the references of a paper. This should not be considered plagiarism, due to the standardized nature of reference sections in scientific papers. Approximately 95% of the overlaps that were detected fell under this category. Besides that, other kinds of false positives include university names, department names, copyright statements, template front pages and other sections of text usually part of a specific format.

Besides that, there were also a few true positives that were found. Whether the overlaps could actually be considered "plagiarism" is up for debate, but the fact is that they are specific sections of text that are suspiciously similar to each other. These true positives were uncommon, and were often only one or two sentences. It makes sense that actual plagiarism only occurs rarely, as it is frowned upon greatly and could get one in a lot of trouble.

The largest overlap score found was 399. This overlap was a reference to a paper with many different authors. This is definitely not plagiarism however, as discussed before. This overlap, and others, can be viewed on the visualisation website.

From Figure 6, we can conclude that the amount of overlapping text sections increased with the growth of the popularity of the internet. Especially the year 2000 seems to be a pivot, the amount of overlapping text has increased massively since then. This could have to do with the fact that the internet makes the papers more accessible. Another possible, and maybe more likely, explanation could be the increase of the amount of online papers in general. If more papers exist, it is likely that the amount of overlapping text also increases.

There are definitely authors who have multiple cases of overlapping text on their name, as visible in Figure 7. The author with the most overlapping text sections has 243 cases. Of course, once again, this does not necessarily imply that they have committed plagiarism 243 times, if at all.

6. DISCUSSION

The goal for this project was to be able to find direct and mosaic plagiarism in a huge data set of scientific research papers. What was achieved, is the ability to find overlapping text sections between papers, possibly with some slight differences. It has proven to be difficult to make a clear distinction between what is plagiarism, and what is not. Some improvements could be made across all phases of the pipeline described in this report.

First of all, in the extraction phase, improvements could be made to the plain text extraction method from the PDF files. The PyPDF2 python package which was used was fast, and often worked very well. However, with a rather large number of papers, it would fail to recognize spaces between words, resulting in a long string of words joined together. How many documents suffered from this issue is hard to tell, since there was no real way to detect if text extraction had suffered from this issue, other than crosschecking every word in the text body with an English dictionary, which would be much too time consuming. Many other packages had similar issues, and packages that did not have these issues were often much too slow to use on such a large data set. This phase could be improved upon by using a better plain text extraction method, which seems to be difficult to come by.

For the cleaning phase, an improvement would be to remove the references sections from the papers. A large number of the false positives the pipeline detected were found in these sections due to their standardized nature. Removing them would increase the accuracy and speed of the pipeline significantly. Other possible text sections to remove would be copyright statements, university names and other section types that are likely to contain overlapping text that would not be considered plagiarism. Sadly, it remains difficult to identify, and thus filter these out of a text body.

The analysis phase is the largest bottleneck of this pipeline in terms of scalability for many reasons. The main method of reducing the number of comparisons is using LDA. By dividing the papers into different topic clusters, only the papers with the same topics need to be compared with each other. The LDA implementation used in the pipeline was very limited in the number of topics that it could create

without throwing 'out of memory' exceptions. The biggest improvement would be to either use an LDA implementation that allows for more clusters to be created, or to use a different clustering algorithm entirely. The ability to create more clusters would significantly reduce the number of comparisons necessary, improving the scalability of the pipeline significantly.

Another possible improvement that could be made is to write the user defined functions throughout the pipeline in Java instead of Python. Python code is generally known to be slow, because it is an interpreted language instead of a compiled one. The best function to apply this change to is the Smith-Waterman function, because the analysis phase is the slowest part of the pipeline. The Smith-Waterman is used a lot in this phase. Optimizing it has a lot of impact on the total time the pipeline takes.

The data set used for this project had many issues. Extracting the plain text from the PDF files correctly took a long time to get right. For example, many PDF files were corrupted. The biggest problem was the fact that there were 11 PDF files in the entire 5 million PDF file data set that caused the workers to freeze indefinitely. No exceptions were thrown, they just stopped working. Finding the PDF files that were causing this problem was like finding a needle in a haystack, and so it set the project back for a week to fix these issues. Because of this, there was barely any time left to implement the improvements mentioned above.

7. REFERENCES

[1] Apache. *Apache Parquet*. `https://parquet.apache.org/`.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] LiteraryTerms. *Plagiarism: Definition and Examples*, September 2020. `https://literaryterms.net/plagiarism/`.

[4] C. U. Press. *Cambridge English Dictionary*, September 2020. `https://dictionary.cambridge.org/dictionary/english/plagiarism`.

[5] Sci-Hub. *Sci-Hub Principles*, September 2020. `https://sci-hub.ren/#principles`.

[6] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[7] Z. Su, B. R. Ahn, K. Y. Eom, M. K. Kang, J. P. Kim, and M. K. Kim. Plagiarism detection using the levenshtein distance and smith-waterman algorithm. *2008 3rd International Conference on Innovative Computing Information and Control*, pages 569–569, June 2008.

[8] Unpaywall. *Data Format*, September 2020. `https://unpaywall.org/data-format`.

[9] M. Wolf and C. Wicksteed. Datetime formatting based on iso 8601. September 1997. `https://www.w3.org/TR/NOTE-datetime`.

8. APPENDIX

Table 1: Task distribution

Who	Tasks
William Ford	Data Extraction, Data Cleaning, Smith Waterman, Visualisation Website, Final Report (Sections 1, 2, 3, 4.2, 4.3.2)
Lars de Jong	Data Analysis, Data Cleaning, LDA implementation, Result graphs & charts, Final Report (Sections 4.3.1, 4.3.3, 4.3.4, 4.3.5, 5, 6)
Max Raams	Data Extraction, Data Analysis, Visualisation, Result analysis, Final Report (Sections 4.1, 4.3.1, 4.3.2)