Data Analytics - Project 1 / Group Number 15 - Suspicious Passages

Christian Altrichter, Francesco Huber

April 12, 2023

1 Introduction

This document outlines the methodology and results for the Data Analytic's Project 1. Here, a group of two students was tasked to apply concepts seen in class to describe and analyse a given data set.

The group consists of the following students:

- Christian Altrichter
- Francesco Huber

Our group has been tasked to conduct a similarity study on the 'suspicious passages' data set. Here, we are tasked to use python and additional libraries to address the following points:

- Data exploration and description
- Pre-processing the data
- Use one duplicate detection algorithm to identify duplicate documents
- Test the accuracy of the model using the test data

Amongst the corpus of documents we were furthermore given a ground truth on which we can compare and evaluate the accuracy of our designed model.

1.1 Submission structure

Our submission zip file contains the following folders / documents:

- DataFrames from core computations that summarize the data (as explained in section 2).
- Modified corpus - Contain the corpora with a reduction in size (explained in section 3).
- Test dir - Contains a smaller test set with 10 documents to test the model in an initial phase (as explained in section 4.2).
- Results - Contains various csv files stored as prediction of the classes. The file names are indicators for what parameters have been used (threshold, k-size and number of permutations). These files can be used to measure the accuracy of our model. These files can be used to with the evaluation.py to test the models against ground truths.
- Computed ground truths - contains the manually computed ground truths as per xml files (explained in section 5.1).

Furthermore, we have worked constantly on github. To see the full repository, please see the below link:

https://github.com/Frakk0/Data_Analytics_Project1.git

2 Data exploration and description

We were given a corpus of 902 documents each varying in length, complexity, and formatting. The ground truth for these documents consists of two classes. The class with suspicious passages (1) and the class with no suspicious passages (0). The split between these two classes is exactly 50.00% each (401 documents class 1 and 401 documents class 0). The summary of the statistics has been done with the pandas.dataframe.describe() method. The documents in its overall composition had the following characteristics:

Document aspect	**Min**	**Max**	**Mean**	**StDev**
Characters	834	2,693,709	171,157	287,009
Unique vocabulary	105	63,248	5,636	7,177
All vocabulary	142	467,280	30,778	51,885
Line counter	13	54,160	3,460	5,702

Table 1: Corpus statistics

As can be seen, the corpus differs in all 4 categories above greatly. The high variety in the size of documents does pose restrictions in the accuracy of duplicate detection algorithms e.g. as outlined in the datasketch documentation for minhash:

https://github.com/ekzhu/datasketch/issues/85

Thus, given the above statistics, going forward we expect to have a more challenging time having a high accuracy compared to the ground truth. In the evaluation part it is crucial to find the parameters that works best for a corpus with a high variety in size.

To analyze the similarity of the documents, we initially set out to find the authors and their publication date respectively. Based on that, we could identify which document has been published in chronological order and therefore infer which document would have been plagiarised from which document. However, as there was not one uniform format for any of the files, a previously written code by us to detect authorship or publication date was inconclusive. Here, we applied multiple methods to filter the document e.g. by the keywords ”produced by” or the word ”author” in the first and last paragraph of the document. Based on the usage of keywords, the results were highly variable. In addition, there is no original source in which we can refer to the author or publication. Therefore, we came to the conclusion that any statistical data on author and publication is meaningless.

2.1 Document Composition

As previously mentioned, the documents are each composed of an individual structure. Thus, we observed that there is no uniformity on the following aspects:

- Placement of table of content
- Placement and presence of document information (e.g. author, publisher, publication date)
- Lower and upper case letters usage (e.g. some titles are all capital letters)
- Type of text (e.g. book, description, listing)
- Content and subject matter

Initially, we intended to cluster the corpus based on the above criteria. Clustering would have helped to more accurately pre-process the documents based on their type without loss of generality. Clustering the documents would most likely produce more accurate results. Due to the variety in the corpus of documents combined with the unclear structure given in each document, we were however not able to cluster the documents accordingly. Thus, we opted for a generic approach to compare all documents and process all documents simultaneously.

Also, the documents have been artificially plagiarised at random. Thus, clustering the documents by any means will lead to the removal of the random links of plagiarism. As a consequence, clustering cannot and should not be performed here.

The corpus of our data set includes both ”txt” and ”xml” files, the former being the full text of the document while the latter representing information about the artificially introduced suspicious passages. Our exploration

of the xml files indicates that plagiarism is very varied. Files can be plagiarised in multiple sections by the same file, also the artificially inserted passage can have been altered according to a variable called ”obfuscation” which corresponds to the level of alteration that the text section undergoes from the original. It is also important to note that the length of the artificially inserted sections varies greatly, from only hundreds of characters to tens of thousands.

3 Data Pre-processing

As the variety in size is expected to have detrimental effects on our model's accuracy, we trimmed each document by smaller paragraphs where the word count is lower than a given mean word count per line. Here, we assume that each text consists of wrapper text passages that do not constitute to the actual content (e.g. table of content, tables, images etc.). Thus, we would reduce the document to its core text passages. Here, we stripped the documents on three different percentages. Based on each percentage, should the size of a line within the text be less than the given percentage of the mean size of a line, then it would be deemed irrelevant and we would not consider it in our computations. Thus, we have 4 corpora after the stripping available to work with:

- The original corpus
- Another three corpora where we remove the lines that are X% smaller than the average char count per line for the entire document.
  - X = 25%
  - X = 50%
  - X = 75%

Subsequently, we have stemmed the vocabulary based on the Porter stemming algorithm using the library ”nltk.stem.porter”. This includes the following pre-processing steps:

- Words to lowercase
- Suffix stripping (e.g. ”ponies” transforms to ”poni”; ”agreed” transforms to ”agree” etc.)

Furthermore, we used the python built-in function ”strip()” to remove all unnecessary white spaces. This reduces the amount of shingles to be produced and it does not add any valuable information to the jaccard similarity computation.

The above pre-processing allows us to analyze data and detect similarity despite having altered the corpus from its original content.

We have not opted to remove special characters due to the nature of the given types of documents present in the corpus. Sometimes, e.g. special characters are used to express a scientific data or passages. Thus, should someone have plagiarised from such, we would not exclude it from its content as this could prove even stronger stronger similarity when computing the k-shingles.

We decided against removing stop words as it could reduce the accuracy of our model. Stop words represent a structural component for our sentences and therefore play an important role in duplicate detection.

It is important to note that any pre-processing of the data must maintain the structural order of the text before being passed to the shingleing step (which we have adhered to).

4 Duplicate detection algorithm - Methodology

As our corpus is very large, we cannot simply compute the jaccard similarity of all documents based on their vocabulary used. This would exceed the computational power of the task. Thus, to detect similar documents, we need to estimate the jaccard similarity to detect whether a document includes suspicious passages that might have been plagiarised. We have followed the approach seen in class. Here, the algorithm to detect similarity has been implemented using the following chronological steps:

- **Step 1 - K-Shingling**: Compute shingles based on a given k. To compute the k shingles, we used the stemmed vocabulary as described before. After having the stemmed vocabulary as list of vocabularies, we

concatenate them back to the original text (but stemmed) to then be able to compute the shingles. Here, we have used the python import "kshingle". This import performs character based shingling to then be fed to the MinHashing and the LSH computations. There are a variety of implementations. It is important to note that we have opted to use the shingles as set and not as bags of shingles. Furthermore, we have defined the shingle set to be of fixed k size.

- **Step 2 - MinHashing**: Here, we used the "datasketch" library, in particular "datasketch.MinHash". Here, the number of permutations will alter of course the computation time accordingly. As we are not dealing with billions of data, we have not opted to used "datasketch.LeanMinHash". Here, we used the regular MinHash implementation to increase our accuracy of the model ultimately.
- **Step 3 - LSH**: Here, we also used the "datasketch" library. The results from the MinHashing step are gathered and compared to deduct the similarity based on maximized probability. This returns the list of most similar documents depending on a threshold or on a given top-k. We have implemented multiple outputs of LSH:
  - LSH output in binary based on a threshold
  - LSH output in a list based on a threshold
  - LSH output in a list based on top-k similar documents

As the computations are respectively quite heavy (on average about 15 minutes per test), we have taken the liberty to state save the computations in csv files.

4.1 Purpose of different code sections - in chronological order of execution

- **core_computation**: Used to obtain statistical data from every document in the set. Populates CSV files containing information such as number of lines, unique vocabularies, the entire vocabulary list, unique characters, all characters and the text as string per document in the source folder.
- **reduce_lines**: Used to reduce the line amount by excluding portions of text which add little to no value to similarity computations. We consider this to be noise within the documents.
- **clean_files_generator**: Used to create folders populated by the intended amount of mean reduced files.
- **similarity_computation _threshold**: Applies all the steps to achieve an LSH analysis, in particular stemming, shingling, minhashing and finally LSH.
- **similarity_computation_top_k**: This section is a variant of similarity that also uses LSH Forests, which aim to provide a ranking list of k top documents that appear to be the most similar to the one in consideration.
- **evaluation**: Works on the produced files of similarity and aims to give a general evaluation on the results by comparing them to the ground truth file. In particular it returns precision, recall and accuracy.
- **clean_ground_truth**: Produces the three documents necessary for a more accurate comparison with our specific data set.
- **ground_truth_evaluation**: evaluates results based on the selected ground truth.
- **Executable**: runs the model specifically to intended threshold amount, shingle size and permutation amount

4.2 Verification of methodology on smaller test-set

To verify whether our algorithm works on a smaller scale, we have created a test directory from which we have taken a total of 10 documents. From this corpus, 6 of the 10 documents are unique. For the remaining 4 documents we have identically copied the files into another txt file. Then we have erased and switched around manually some text passages to accurately test our algorithms performance. Amongst the 10 documents, 3 documents have no similarity with any other document. The remaining 7 have similarities with other txt files and therefore should be flagged as plagiarised.

The output of the LSH is a list of documents that pass the threshold from the LSH. It is trivial that the document will find itself as similar document. The other document is a document that is then (according to the threshold) similar. Then we transform that output into a binary data frame to be able to compare the results to the ground truth. We do this by testing, whether the length of the list of similar documents is greater than 1 (because of the inclusion of itself). If so, a document with a suspicious passage has been identified and marked as plagiarised.

To test this, we have used a threshold of 0.3, a shingle size of k = 8 and 128 number of permutations. Figure 1 shows the output for the test set of our algorithm:

```
------------------------------------------------------------
DATAFRAME BASED ON THRESHOLD - LIST
TestFile -> ['TestFile1', 'TestFile', 'TestFile3']
TestFile1 -> ['TestFile1', 'TestFile', 'TestFile3']
TestFile2 -> ['TestFile2']
TestFile3 -> ['TestFile', 'TestFile1', 'TestFile3']
TestFile4 -> ['TestFile4', 'TestFile5']
TestFile5 -> ['TestFile4', 'TestFile5']
TestFile6 -> ['TestFile7', 'TestFile6']
TestFile7 -> ['TestFile7', 'TestFile6']
TestFile8 -> ['TestFile8']
TestFile9 -> ['TestFile9']
------------------------------------------------------------
DATAFRAME BASED ON THRESHOLD - BINARY
Document, Plagiarised
TestFile  ->  1
TestFile1  ->  1
TestFile2  ->  0
TestFile3  ->  1
TestFile4  ->  1
TestFile5  ->  1
TestFile6  ->  1
TestFile7  ->  1
TestFile8  ->  0
TestFile9  ->  0
------------------------------------------------------------
Per document application, query of 1st document
REGULAR LSH: ['TestFile1', 'TestFile', 'TestFile3']
forest of top k elements close to 1st document
e.g.: k=2
FOREST LSH: ['TestFile1', 'TestFile']
```

Figure 1: Output from the terminal based on our test set

Below the summary of the initial findings:

Document nr.	**Similar passages as**	**Predicted plagiarism from**	**Ground truth**	**Predicted**
Document nr. 0	Document nr. 3 and 1	Document nr. 3 and 1	1	1
Document nr. 1	Document nr. 3 and 0	Document nr. 3 and 0	1	1
Document nr. 2	None	None	0	0
Document nr. 3	Document nr. 1 and 0	Document nr. 1 and 0	1	1
Document nr. 4	Document nr. 5	Document nr. 5	1	1
Document nr. 5	Document nr. 4	Document nr. 4	1	1
Document nr. 6	Document nr. 7	Document nr. 7	1	1
Document nr. 7	Document nr. 6	Document nr. 6	1	1
Document nr. 8	None	None	0	0
Document nr. 9	None	None	0	0

Table 2: Plagiarism summary, ground truth and prediction of manual computed test set

For the results we have the following anaysis:

- Precision: 100.00%
- Recall: 100.00%
- Accuracy: 100.00%

We have realized that however to be really accurate in detecting suspicious passages a text does not need to contain a sizeable portion of the original text. In theory even a couple lines out of hundreds should constitute as suspicious passages of sorts. Therefore we further tested this aspect by creating a new file called TestFileZ which is duplicate of TestFile8 with 2 lines from TestFile0. Our initial thresholds and shingle size were not able to pick up these matching passages, but by lowering considerably our threshold from 0.6 to 0.01 and increasing the shingle size from 4 to 20, we were finally able to pick up the difference while preserving the unique documents (the ones not similar to others) non similarity to the rest of the test set.

5 Results

5.1 Testing with given ground truth

The results of our model are in a tradeoff with the following parameters:

- Shingle size
- Number of permutations (thus, number of MinHash functions)
- Threshold

For the following explanation, as the classes are equally balanced, we choose the accuracy as main indicator for the performance of our model. Initially, we started experimenting with the threshold. Here, we have gotten the following results with varying threshold:

Shingle Size	**# perm**	**Threshold**	**Precision [in %]**	**Recall [in %]**	**Accuracy [in %]**
8	128	0.2	50.31	90.47	50.55
8	128	0.25	51.52	45.23	51.33
8	128	0.3	51.43	43.9	51.22
8	128	0.35	43.33	8.65	48.67
8	128	0.4	43.33	8.65	48.67
8	128	0.45	33.33	0.44	49.78
8	128	0.5	33.33	0.44	49.78

Table 3: Precision, Recall, and Accuracy computed on given ground truth with varying threshold.

As can be observed in the table above (table 3), we have achieved the best result with a threshold of 0.25 in terms of accuracy. However, the recall is slightly higher compared to a threshold of 0.3, thus, we assume the accuracy is high only by ratio of documents that incorporate suspicious passages. Thus, we have taken taken the threshold of 0.3 (with a lower recall) and varied the size of the shingle sets as follows:

Shingle Size	**# perm**	**Threshold**	**Precision [in %]**	**Recall [in %]**	**Accuracy [in %]**
3	128	0.3	50.00	100.00	50.00
4	128	0.3	50.17	99.78	50.33
5	128	0.3	50.69	97.56	51.33
6	128	0.3	51.40	85.59	52.33
7	128	0.3	51.91	72.28	52.66
10	128	0.3	50.65	8.65	50.11
12	128	0.3	50.00	100.00	50.00

Table 4: Precision, Recall, and Accuracy computed on given ground truth with varying shingle size.

Similar to the initial step, we have taken a look at precision, recall and accuracy and have decided that the best results are performed on a shingle size of 6. Thus, we have taken the shingle size of 6 and threshold of 0.3 and have adjusted the number of permutations. This leads us to the following results:

An additional phase of testing was carried out onto the documents that had been processed in relation to their average character length per line. We have performed various reductions according to the aforementioned data pre-processing (3) step to reduce document specific passages (e.g. tables, introductions etc.). Below, we have tested on a 25%, 50% and 75% reduction.

Shingle Size	# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy [in %]
6	256	0.3	49.35	59.20	49.22
6	512	0.3	51.65	72.73	52.33
6	1024	0.3	51.90	51.44	51.88
6	2048	0.3	51.49	57.65	51.66

Table 5: Precision, Recall, and Accuracy computed on given ground truth with varying number of permutations.

Reduction	Shingle Size	# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy
25%	6	128	0.3	51.12	90.91	52.00
25%	12	128	0.3	43.75	3.10	49.56
50%	6	128	0.3	51.18	91.13	52.11
50%	12	128	0.3	44.12	3.33	49.56
75%	6	128	0.3	51.11	91.80	52.00
75%	12	128	0.3	51.61	3.55	50.11

Table 6: Precision, Recall, and Accuracy computed on given ground truth with reduction of passages based on mean word count per line.

Since we haven't seen any major increase in the above tested variations, we now believed that, as mentioned already initially by the datasketch documentation, the corpus varies in size too much. Hence, we attempted to reduce the corpus size (e.g. by 1/3) and test the model on that. The splitting was done at random and the results are as following:

Corpus reduction	Shingle Size	# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy
66.66%	6	128	0.30	49.77	78.01	51.03

Table 7: Precision, Recall, and Accuracy computed on given ground truth with only one-third of the corpus selected at random

A further aspect we tested on in which we hoped for some improvements on our model was the use of unigrams (thus, tokens being words instead of characters instead of fixed size shingles), based on the assumption that the vocabulary similarity could have helped with such high dimension data set. This however did not produce any meaningful results which are listed in the table below.

# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy
128	0.50	51.55	44.35	51.33

Table 8: Precision, Recall, and Accuracy computed on given ground truth with unigrams instead of shingles

After having tested almost every possible parameter permutation, we intended to extract two documents and all its similar documents with artificially created insertions. Here, we had a look into the xml files and test our model on this data set. Any permutation above did not change in accuracy which lead us to believe that there was an error within our model. However, this is contradictory to the manually created test set that performed perfectly. Thus, when taking a closer look at two documents ("document00634" & "document00014") we have found the following observation:

5.2 Ground Truth Discrepancies

Given that our accuracy did not improve over the multiple iterations of testings, we started investigating on the provided ground truth and the indicated sources of plagiarism present on the xml files.

First, for document00634 we have found that the document is mentioned in the ground truth as plagiarised (thus, class 1). However, all documents from which it shares suspicious passages were not found to be within the our corpus:

Similarly, for the document00014 we have the following discrepancies as another example with our corpus and the plagiarised files. This document however was correctly labeled as plagiarised.

Similar to document	Document present in corpus
Document02492	No
Document05765	No

Table 9: Documents that are supposed to share suspicious passages with document00634 and with their respective presence in the provided corpuse

Similar to document	Document present in corpus
Document00852	Yes
Document00912	No
Document02701	No
Document03390	No
Document03686	No
Document04633	No
Document04678	Yes
Document05507	No
Document05760	No
Document05822	No
Document06355	No

Table 10: Documents that are supposed to share suspicious passages with document00014 and with their respective presence in the provided corpuse

For the document "document00014" we have detected with shingle size 6, threshold 0.1 and number of permutations 128 that the document in fact shares suspicious passages. The document tree detects that all documents listed above **and** are present in the corpus were detected.

Thus, we conclude that the inconsistency and mismatch between ground truth and the actually available documents from with which a document supposedly shares suspicious passages within the corpus lead to a misleading accuracy computation of our model. This explains why any alteration we have done in terms of hyperparameter tuning has not affected the model in a meaningful way. The given corpus is therefore not entirely accurate when it comes to detecting actual duplicates. As previously already shown in the verification step of our methodology, our model works perfectly at a smaller scale.

When recomputing the ground truth according to Case 1, we have found that the distribution is uneven. Now, we have approximately 15.00% of class 1 and the remainder of class 0. When recomputing the ground truth according to Case 2, we have approximately 30.00% of class 1 and the remainder of class 0. Thus, the metrics we computed prior (precision, recall and accuracy) does not prove to be effective and indicative anymore as the classes are imbalanced. Therefore, we will refer to metrics that are better suited when the positive class is in a minority distribution. Here, we can typically use the F1 score and plot the ROC-Curve to show the accuracy of our model.

In trying to show the real effectiveness of our model, we have scanned through each xml file and created an actual ground truth based on the documents that it has taken passages from. We created three documents that we used to gain access to more meaningful comparison methods and increase our analysing capabilities of our data set:

1. **"actual_ground_truth.tsv"** is the revised ground truth with plagiarism only indicated for the files that display artificial insertions by existing files in our data set (e.g. as described in section 5.1 - Case 1).
2. **"speculative_ground_truth.tsv"** aims to provide similar results as the actual ground truth, but considers the bi-implication of a suspicious passage in one document (e.g. as described in section 5.1 - Case 2).
3. **"plagiarism_links.tsv"** expands on the previous document by indicating concretely what file sources are supposed to be the origin of the artificial insertions. Thus, the more documents link to one document, we will most likely be able to pinpoint to the origin of all documents. However, as the insertions are of artificial nature, we the link structure will equally be random.

Based on these three files we have taken two approaches to compute the actual ground truth:

- Case 1 (actual ground truth): If a document has a list of documents in the xml files it shares passages with **and** the document(s) within this list are present within our corpus are present, then we flag this document as class 1. Based on this principal the initial ground truth has also been computed.
- Case 2 (speculated ground truth): However, the above list also implies that there is a binary relationship between the documents. Thus, if one document shares passages with another document, then both can be in another scenario in class 1 (thus, both marked as suspicious passages). Our model is unable to detect which document plagiarised from which. However, our speculative ground truth can classify both documents as 1.

We now need to test our model for both cases mentioned above. We expect that our model will behave better in Case 2.

5.3 Testing with computed ground truth of Case 1 and Case 2

Given that our hyper parameter tuning was ineffective prior, we have now started to recompute the F1 score for the ground truth based on case 1 and case 2. Here, our aim is now to continue with the highest F1 score based on our previous computations and then tune the hyper parameters again. We started off with the following parameters as it achieved the overall best performance of all previous computations in terms of F1 score for Case 2. Here, we achieved a max F1 score with a 0.1 threshold, shingle size of 8 and number of permutations of 128. The max F1 score as initial benchmark was at 45.45%. We then subsequently altered the shingle size to observe any viable altercations:

Shingle Size	# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy [in %]	F1 [in %]
8	128	0.1	15.18	94.16	19.18	26.14
10	128	0.1	14.17	64.23	35.48	23.22
12	128	0.1	14.41	36.50	57.43	20.66
14	128	0.1	11.29	10.22	74.17	10.73
16	128	0.1	14.75	6.57	80.04	9.09

Table 11: Results based on comparison with **Case1** ground truth.

Shingle Size	# perm	Threshold	Precision [in %]	Recall [in %]	Accuracy [in %]	F1 [in %]
8	128	0.1	30.00	93.75	31.15	45.45
10	128	0.1	28.99	66.18	40.91	40.31
12	128	0.1	29.97	38.24	54.43	33.60
14	128	0.1	25.00	11.40	62.97	15.66
16	128	0.1	27.87	6.25	66.85	10.21

Table 12: Results based on comparison with **Case2** ground truth.

We assume one of the reasons for of a lack of better results in comparison to any ground truth, is linked to the obfuscation level and size of the copied suspicious passage of the artificially inserted sections. As previously stated our technique does not work well for passages that have been considerably altered from the original, due to the time constraints and previous extensive testing we were unable to examine further on the effective validity of our assumptions and find a model that fits best the data set.

6 Conclusion

In conclusion we carried out an analysis of the ”suspicious passages data set”, our work allowed us to complete following given tasks:

- Explore and describe the data
- Pre-process the data (i.e. handle and fill unknowns if there are any etc.)
- Use one duplicate detection algorithm to identify duplicate documents.
- Test the accuracy of the model using the test data.

As shown by our tests, the model we devised based on the LSH approach performed the task well of detecting suspicious passages amongst documents. However, when initially applying the model to the general data set, we were not able to reach conclusive results based on the given ground truth. This prompted us to apply a more systematical approach to better tweak our parameters, namely our threshold amount, shingle size and number of permutations. After thorough and time intensive testing and little improvement, we focused our attention on the validity of the given ground truth and found that many of the files that were supposed to impact the determination of the given classes were simply missing from our data set. Thus, effectively rendering our previous comparisons inconsequential. In order to have a clearer idea on the validity of our model we computed our own ground truth based off of the existing files in the data set and the information available within the xml files as described in section 5.2.

Going forward we recommend further testing on the impact that levels of alteration of artificially inserted passages in our data set play on the results of our model. This should undergo the rigorous process that we have applied initially to the ground truth to find the sweet spot for an accurate model. However, this requires further time intensive testing and hyper parameter tuning.

7 URL Packages

external libraries:
pandas: https://pandas.pydata.org/
csv: https://github.com/python/cpython/blob/3.11/Lib/csv.py
re: https://github.com/python/cpython/tree/3.11/Lib/re/
string: https://github.com/python/cpython/blob/3.11/Lib/string.py
numpy: https://numpy.org/
datasketch https://pypi.org/project/datasketch/
nltk: https://www.nltk.org/

graphing external libraries:
networkx: https://networkx.org//
matplotlib: https://matplotlib.org///