

Homework 1

Finding Similar Items: Textually Similar Documents

Badai Kesuma

Hamid Dimyati

November 2020

1. Code commentary

The implementation of the task is done in Python mainly using numpy, pandas, and nltk libraries. The application finds the similarity of 100 documents from “*A Corpus of Plagiarised Short Answers*” data set¹. This application is capable of extracting pairs of similar documents from a data set presented in that corpus. Documents containing similar texts are shown in the hash similarity score and signature similarity score. Jaccard similarity is used as a similarity measure to find identical pairs.

Within the code (`lsh.py`), we created a class called `shingling` which requires the parameter `k` as the number of words in one shingle. This class gives an output in the form of Jaccard similarity of each pair of documents. Apart from this matrix, we also save a Characteristic matrix within the class object. The next one, we created a class called `minhashing` which processes the Characteristic matrix into smaller matrices, Signature matrix and saves it into the class object. This class also requests parameter `k` as the number of signatures. The output of this class would be the Signature score for each pair of documents. Lastly, we implemented a class called `lsh` to run the Locality Sensitive Hashing (LSH) by processing the Signature matrix to get a set of candidates that meet the `threshold` parameter. Apart from `threshold`, this class also requires parameter `bands` as number of bands and `rows` as number of rows in each band.

¹ https://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

2. How to run

```
git clone https://github.com/hamiddimyati/id2222-data-mining-advanced.git
cd id2222-data-mining-advanced/assignment-1
python3 lsh.py
```

We could also modify all the parameters through the python code, such as `k`, `k_signature`, `bands`, `rows`, and `threshold`.

3. Results

In order to see the results, we have to look at the hash similarity score and signatures similarity score from each pair of documents. In hash similarity score, we see the number of similar shingles divided by the number of total shingles which both documents do not equal to zero. In signatures similarity score, we see the number of similar signatures divided by the number of all signatures.

From the result of Figure 1 we found that many pairs of documents have a similarity score of zero in both hash similarity score and signatures similarity score, which means that the pair of documents are completely different. From Figure 2 we see the pair of documents which have higher scores in the similarity, which indicate that the documents are plagiarized. The shingles length and the number of signatures affect the output of the similarity scores. Both hash similarity scores and signatures similarity scores give almost the same results which mean the approximation in signatures give good results compared to the exact computation in the hash, which was what we expected.

Figure 3 and Figure 4 are the results from the application (`lsh.py`), which are the most similar documents based on Signature similarity score and the sample of pair of candidates based on LSH, respectively.

Doc1	Doc2	Jaccard_Score	Doc1	Doc2	Jaccard_Score	Doc1	Doc2	Signature_Score	Doc1	Doc2	Signature_Score
0	Doc0	Doc1	0.000000	0.00
1	Doc0	Doc2	0.000000	9870	Doc99	Doc69	0.172794	1	Doc0	Doc2	0.00
2	Doc0	Doc3	0.000000	9871	Doc99	Doc70	0.000000	2	Doc0	Doc3	0.00
3	Doc0	Doc4	0.000000	9872	Doc99	Doc71	0.000000	3	Doc0	Doc4	0.00
4	Doc0	Doc5	0.000000	9873	Doc99	Doc72	0.000000	4	Doc0	Doc5	0.00
5	Doc0	Doc6	0.000000	9874	Doc99	Doc73	0.000000	5	Doc0	Doc6	0.00
6	Doc0	Doc7	0.000000	9875	Doc99	Doc74	0.018613	6	Doc0	Doc7	0.00
7	Doc0	Doc8	0.000000	9876	Doc99	Doc75	0.000000	7	Doc0	Doc8	0.00
8	Doc0	Doc9	0.000000	9877	Doc99	Doc76	0.000000	8	Doc0	Doc9	0.00
9	Doc0	Doc10	0.000000	9878	Doc99	Doc77	0.000000	9	Doc0	Doc10	0.00
10	Doc0	Doc11	0.000000	9879	Doc99	Doc78	0.000000	10	Doc0	Doc11	0.00
11	Doc0	Doc12	0.000000	9880	Doc99	Doc79	0.481481	11	Doc0	Doc12	0.00
12	Doc0	Doc13	0.000000	9881	Doc99	Doc80	0.000000	12	Doc0	Doc13	0.00
13	Doc0	Doc14	0.000000	9882	Doc99	Doc81	0.000000	13	Doc0	Doc14	0.00
14	Doc0	Doc15	0.000000	9883	Doc99	Doc82	0.000000	14	Doc0	Doc15	0.00
15	Doc0	Doc16	0.000000	9884	Doc99	Doc83	0.000000	15	Doc0	Doc16	0.00
16	Doc0	Doc17	0.000000	9885	Doc99	Doc84	0.241379	16	Doc0	Doc17	0.00
17	Doc0	Doc18	0.000000	9886	Doc99	Doc85	0.000000	17	Doc0	Doc18	0.00
18	Doc0	Doc19	0.000000	9887	Doc99	Doc86	0.000000	18	Doc0	Doc19	0.00
19	Doc0	Doc20	0.000000	9888	Doc99	Doc87	0.000000	19	Doc0	Doc20	0.00
20	Doc0	Doc21	0.000000	9889	Doc99	Doc88	0.000000	20	Doc0	Doc21	0.00
21	Doc0	Doc22	0.000000	9890	Doc99	Doc89	0.079023	21	Doc0	Doc22	0.00
22	Doc0	Doc23	0.000000	9891	Doc99	Doc90	0.000000	22	Doc0	Doc23	0.00
23	Doc0	Doc24	0.000000	9892	Doc99	Doc91	0.000000	23	Doc0	Doc24	0.00
24	Doc0	Doc25	0.000000	9893	Doc99	Doc92	0.000000	24	Doc0	Doc25	0.00
25	Doc0	Doc26	0.000000	9894	Doc99	Doc93	0.000000	25	Doc0	Doc26	0.00
26	Doc0	Doc27	0.000000	9895	Doc99	Doc94	0.002725	26	Doc0	Doc27	0.00
27	Doc0	Doc28	0.000000	9896	Doc99	Doc95	0.000000	27	Doc0	Doc28	0.00
28	Doc0	Doc29	0.000000	9897	Doc99	Doc96	0.000000	28	Doc0	Doc29	0.00
29	Doc0	Doc30	0.000000	9898	Doc99	Doc97	0.000000	29	Doc0	Doc30	0.00
...	9899	Doc99	Doc98	0.000000

3900 rows x 3 columns

9900 rows x 3 columns

Figure 1. The output of hash similarity scores on each pair of documents (left) and signature similarity scores on each pair of documents (right)

	Doc1	Doc2	Jaccard_Score	Signature_Score
9425	Doc95	Doc20	0.878689	1.00
2074	Doc20	Doc95	0.878689	1.00
9485	Doc95	Doc80	0.790625	0.75
8014	Doc80	Doc95	0.790625	0.75
7940	Doc80	Doc20	0.683891	0.75
2059	Doc20	Doc80	0.683891	0.75
9765	Doc98	Doc63	0.619048	1.00
6334	Doc63	Doc98	0.619048	1.00
9785	Doc98	Doc83	0.493438	0.75
8314	Doc83	Doc98	0.493438	0.75
7919	Doc79	Doc99	0.481481	0.50
9880	Doc99	Doc79	0.481481	0.50
8280	Doc83	Doc63	0.459948	0.75
6319	Doc63	Doc83	0.459948	0.75
6855	Doc69	Doc24	0.366460	0.25
2444	Doc24	Doc69	0.366460	0.25
4949	Doc49	Doc99	0.357394	0.50
9850	Doc99	Doc49	0.357394	0.50
2049	Doc20	Doc70	0.341216	0.00
6950	Doc70	Doc20	0.341216	0.00
7009	Doc70	Doc80	0.341216	0.25
7990	Doc80	Doc70	0.341216	0.25
9610	Doc97	Doc7	0.340050	0.25
789	Doc7	Doc97	0.340050	0.25
9475	Doc95	Doc70	0.320635	0.00
7024	Doc70	Doc95	0.320635	0.00
5029	Doc50	Doc80	0.316667	0.50
7970	Doc80	Doc50	0.316667	0.50
5044	Doc50	Doc95	0.300792	0.50
9455	Doc95	Doc50	0.300792	0.50

Figure 2. The output of similarity scores > 0.3

```
(base) Hamids-MacBook-Pro:assignment-1 hamiddimyati$ python3 lsh.py
Load data success! in 0.06 secs
Shingling success! in 26.85 secs
MinHashing success! in 76.16 secs
Most Similar Documents:
Jaccard Similarity: 0.7440476190476191
Signature Similarity: 1.0
Document 1:
In probability theory, Bayes' theorem (often called Bayes' law after Rev Thomas Bayes) relates the conditional and marginal probabilities of two random events. It is often used to compute posterior probabilities given observations (for example, a patient may be observed to have certain symptoms). Bayes' theorem can be used to compute the probability that a proposed diagnosis is correct, given that observation. As a formal theorem, Bayes' theorem is valid in all common interpretations of probability. However, it plays a central role in the debate around the foundations of statistics: frequentist and Bayesian interpretations disagree about the ways in which probabilities should be assigned in applications. Frequentists assign probabilities to random events according to their frequencies of occurrence or to subsets of populations as proportions of the whole, while Bayesians describe probabilities in terms of beliefs and degrees of uncertainty. The articles on Bayesian probability and frequentist probability discuss these debates in greater detail. Bayes' theorem relates the conditional and marginal probabilities of events A and B, where B has a non-vanishing probability:  $P(A|B) = (P(B|A) \times P(A)) / P(B)$ . Each term in Bayes' theorem has a conventional name:  $P(A)$  is the prior probability or marginal probability of A. It is "prior" in the sense that it does not take into account any information about B.  $P(A|B)$  is the conditional probability of A, given B. It is also called the posterior probability because it is derived from or depends upon the specified value of B.  $P(B|A)$  is the conditional probability of B given A.  $P(B)$  is the prior or marginal probability of B, and acts as a normalizing constant. Intuitively, Bayes' theorem in this form describes the way in which one's beliefs about observing 'A' are updated by having observed 'B'.
Document 2:
In probability theory, Bayes' theorem (often called Bayes' law after Rev Thomas Bayes) relates the conditional and marginal probabilities of two random events. It is often used to compute posterior probabilities given observations. For example, a patient may be observed to have certain symptoms. Bayes' theorem can be used to compute the probability that a proposed diagnosis is correct, given that observation. (See example 2) As a formal theorem, Bayes' theorem is valid in all common interpretations of probability. However, it plays a central role in the debate around the foundations of statistics: frequentist and Bayesian interpretations disagree about the ways in which probabilities should be assigned in applications. Frequentists assign probabilities to random events according to their frequencies of occurrence or to subsets of populations as proportions of the whole, while Bayesians describe probabilities in terms of beliefs and degrees of uncertainty. The articles on Bayesian probability and frequentist probability discuss these debates in greater detail. Bayes' theorem relates the conditional and marginal probabilities of events A and B, where B has a non-vanishing probability:  $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$ . Each term in Bayes' theorem has a conventional name:  $P(A)$  is the prior probability or marginal probability of A. It is "prior" in the sense that it does not take into account any information about B.  $P(A|B)$  is the conditional probability of A, given B. It is also called the posterior probability because it is derived from or depends upon the specified value of B.  $P(B|A)$  is the conditional probability of B given A.  $P(B)$  is the prior or marginal probability of B, and acts as a normalizing constant. Intuitively, Bayes' theorem in this form describes the way in which one's beliefs about observing 'A' are updated by having observed 'B'.
```

Figure 3. Output of minhashing class as the most similar documents based on Signature similarity score

```
Locality Sensitive Hashing success! in 43.93 secs
Sample Pair of Candidates:
Document 1:
In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula. The new classes, known as derived classes, take over (or inherit) attributes and behavior of the pre-existing classes, which are referred to as base classes (or ancestor classes). It is intended to help reuse existing code with little or no modification. Inheritance provides the support for representation by categorization in computer languages. Categorization is a powerful mechanism number of information processing, crucial to human learning by means of generalization (what is known about specific entities is applied to a wider group given a belongs relation can be established) and cognitive economy (less information needs to be stored about each specific entity, only its particularities). Inheritance is also sometimes called generalization, because the is-a relationships represent a hierarchy between classes of objects. For instance, a "fruit" is a generalization of "apple", "orange", "mango" and many others. One can consider fruit to be an abstraction of apple, orange, etc. Conversely, since apples are fruit (i.e., an apple is-a fruit), apples may naturally inherit all the properties common to all fruit, such as being a fleshy container for the seed of a plant. An advantage of inheritance is that modules with sufficiently similar interfaces can share a lot of code, reducing the complexity of the program. Inheritance therefore has another view, a dual, called polymorphism, which describes many pieces of code being controlled by shared control code. Inheritance is typically accomplished either by overriding (replacing) one or more methods exposed by ancestor, or by adding new methods to those exposed by an ancestor. Complex inheritance, or inheritance used within a design that is not sufficiently mature, may lead to the Yo-yo problem.
Document 2:
In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula. Inheritance provides the support for representation by categorization in computer languages. Categorization is a powerful mechanism number of information processing, crucial to human learning by means of generalization and cognitive economy (less information needs to be stored about each specific entity, only its particularities). The new classes, known as derived classes, take over (or inherit) attributes and behavior of the pre-existing classes, which are referred to as base classes (or ancestor classes). It is intended to help reuse existing code with little or no modification. Inheritance is also sometimes called generalization, because the is-a relationships represent a hierarchy between classes of objects. For instance, a "fruit" is a generalization of "apple", "orange", "mango" and many others. One can consider fruit to be an abstraction of apple, orange, etc. Conversely, since apples are fruit (i.e., an apple is-a fruit), apples may naturally inherit all the properties common to all fruit, such as being a fleshy container for the seed of a plant. An advantage of inheritance is that modules with sufficiently similar interfaces can share a lot of code, reducing the complexity of the program. Inheritance therefore has another view, a dual, called polymorphism, which describes many pieces of code being controlled by shared control code. Inheritance is typically accomplished either by overriding (replacing) one or more methods exposed by ancestor, or by adding new methods to those exposed by an ancestor. Complex inheritance, or inheritance used within a design that is not sufficiently mature, may lead to the Yo-yo problem.
```

Figure 4. Output of lsh class as the sample of pair of candidates to be further checked

References

- [1] Clough, P., & Stevenson, M. (2011). Developing a corpus of plagiarised short answers. *Language resources and evaluation*, 45(1), 5-24.