Text Technologies for Data Science Assignment 3

s1140740

November 3, 2014

Introduction

The purpose of this report is to provide an overview of decisions made in designing a plagiarism detection tool and cover different types of plagiarisms detected.

Plagiarism types

Type 1 Plagiarisms of this type are exact duplicates, e.g. documents `t104` and `t4172` which are classified as type 1 in `type1.truth` are exactly the same, word for word.

Type 2 Plagiarisms of this type are near duplicates, e.g. documents `t1088` and `t5015` which are classified as type 2 in `type2.truth` are almost the same, except document `t1088` is missing the word *room*.

Plagiarism Detection

The Simhash algorithm together with the MD5 hashing function were used to detect duplicates. For each document an MD5 hash is computed and the document is then tokenized on white-space and non-alphanumeric characters with stop words removed.[1]

Furthermore, a 128-bit fingerprint is computed using the Simhash algorithm with MD5 as a hashing function and term frequencies as weights. This fingerprint is then split into L chunks of size K and the i-th chunk is used as a key of a bucket containing documents having the same chunk in the i-th hash-table. When two documents end up in the same bucket of a hash-table we treat them as possible duplicates.

[1] List of English stop words obtained from `https://github.com/Alir3z4/stop-words` by Alireza Savand. No changes made.

Type 1 detection

To detect plagiarisms of type 1, an MD5 hash of a document is compared to every document in the same buckets. If these hashes are identical then we flag the documents as exact duplicates (type 1). Note, that two stories that are identical are guaranteed to end up in the same buckets as their vector space representation and thus their Simhash fingerprints are the same.

100% precision and recall compared to the file `type1.truth` were achieved using this technique.

Type 2 detection

To detect plagiarisms of type 2, a cosine similarity measure using term frequency weights is computed for a document and all the other documents in the same buckets. If this measure is above a certain threshold, we flag the documents as near duplicates.

Choosing the cosine similarity threshold was challenging as the minimum similarity for a pair of plagiarisms in `type2.truth` was ~ 0.999. However, setting this threshold so low could have negative effects on recall. Thus, the threshold was set to 0.9 as this does not lower precision on documents in `type2.truth` and it gives a reasonable bit of freedom for plagiarism deviation.

Experiments were ran on different values of L and K. With $L = 8$ and $K = 16$, 100% precision and recall compared to the file `type2.truth` were achieved. With $L = 16$ and $K = 8$, too many documents ended up in same buckets and the processing took much longer. With $L = 4$ and $K = 32$, recall was only 90% for plagiarisms of type 2.

Hashing functions SHA-1, SHA-256 and SHA-512 which have digest sizes larger than 16 bytes were benchmarked as well but due to their slower performance[2] they were abandoned in favour of MD5.

[2] Performance of Python `hashlib` hash functions `https://gist.github.com/1992443dc5a7b26c24c6`

Type 3 detection

To detect plagiarisms of type 3, all areas with high number densities are extracted from the data file using Finn's method implemented with an $O(n)$ algorithm. Extracts from Finn's method smaller than 4 tokens are discarded as per task 5 description. Then the same algorithms as for plagiarisms of types 1 and 2 are used to detect duplicates of type 3.

This way, 13 duplicates were detected in the dataset `data.finn`.