Building an inverted index

Language Technology

Miquel Puig i Mena

September 2019

1 Objective

Build a master-index from the corpus in each individual of a set of documents. Represent each document using the TFIDF value and compute similarity between them.

2 Motivation

Nowadays, enormous interest on written document processing has raised. In an automated era as in now, very well known application such as plagiarism detection, web search engines or automatic essay grading apply document processing techniques.

A common paradigm for each application is how to compare documents, in other words, how to be highly confident whether a text is related to another one or not. This report tries attack this problem and provides a feasible solution.

3 Implementation

Extensive set of documents (dataset) might end up in higher system's efficiency since it provides a better representation of the language. Nevertheless, for document comparison, the ability to generalize a concrete language is not the key factor seeing that document analogy is implemented relative to the files being compared.

Selma Lagerlöf's set of novels will be used, meaning that from a group of novels from the same author, the system we'll try to find the most related ones under a human perception.

3.1 Building Master-Index

Archives have to be transformed so they can be compared to each other. Transformation consists in creating a so-called index representation for each document. Index contains a list of all contained words in the file with the numerical indexes where each word occurs. Note that an index will never contain equal word entrances; instead, it will extend the appearances list of hypothetical word.

Extending the idea, a **master-index will include a unification of each index created during the process**. *populate_master_index_by_file* function populates master-index dictionary skipping creation of individual indexes.

```
class Indexer:
    # ...
    def populate_master_index_by_file(self, file_name):
        file = open(SELMA_PATH + file_name, encoding='UTF8')
                    .read()
                    .lower()
        words_in_file = re.finditer('\p{L}+', file)
        for iteration, word_appearances in enumerate(words_in_file):
            self.MI.persist_dict(file_name,
                                 word_appearances.group(),
                                 word_appearances.start())
class MasterIndex:
    # ...
    def persist_dict(self, file_name, word, start_position):
        if word in self.myDict:
            if file_name in self.myDict[word]:
                self.myDict[word][file_name].append(start_position)
            else:
                self.myDict[word].update({file_name: [start_position]})
        else:
            self.myDict[word] = {file_name: [start_position]}
```

3.2 Term frequency Inverse document frequency (TF-IDF) Document Representation

In information retrieval, tfidf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus[Wik]. TFIDF metric can be represented as product of two factors Term Frequency (TF) and Inverse Document Frequency (IDF).

$$TF = \frac{word\_repetitions}{self.files\_length[file\_name]} \tag{1}$$

$$IDF = \log_{10} \frac{len(self.files\_length)}{document\_appearances} \tag{2}$$

Where IDF discriminates words that are seen in all files such as conjunctions, prepositions, etc.

3.3 Document Similarity

Reached this point, the system is capable to retrieve a tfidf representation of each file, which provides a numerical value expressing the importance of each word in a document. Finally, we can compute distance between two TFIDF representations by applying Cosine Similarity. See Table 3.3.

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space (See eq. 3). The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together[Pra].

$$cos(\theta) = \frac{\sum_{i=1}^{n} a_i * b_i}{\sqrt{\sum_{i=1}^{n} a^i} * \sqrt{\sum_{i=1}^{n} b^i}} \tag{3}$$

	bannlyst	gosta	herrgard	jerusalem	kejsaren	marbacka	nils	osynliga	troll
bannlyst		0.005229	0.001299	0.007318	0.002145	0.005096	0.005343	0.005984	0.007336
gosta	0.005229		0.015142	0.008540	0.006509	0.028561	0.025006	0.055747	0.032494
herrgard	0.001299	0.015142		0.007622	0.000892	0.011697	0.014146	0.021256	0.003986
jerusalem	0.007318	0.008540	0.007622		0.00206	0.012904	0.01308	0.041844	0.007064
kejsaren	0.002145	0.006509	0.000892	0.00206		0.044558	0.004374	0.005494	0.088312
marbacka	0.005096	0.028561	0.011697	0.012904	0.044558		0.040214	0.042106	0.023648
nils	0.005343	0.025006	0.014146	0.01309	0.004374	0.040214		0.032128	0.019387
osynliga	0.005984	0.055747	0.021256	0.041844	0.005494	0.042106	0.032128		0.028266
troll	0.007336	0.032494	0.003986	0.007064	0.088312	0.023648	0.019387	0.028266	

4 Conclusion

It's been achieved a document analyser capable of assigning numeric value for similarity between documents. After tests performed, **the system defines that the documents troll.txt and kejsaren.txt are the most similar ones**, therefore, they compute biggest cosine similitude using their tfidf representation.

Finally, such technique explained during the report can be used in real applications. We can observe that the index encoding used in the system contains the same idea as the one used by Google and explained by Jeff Dean in [Dea]. To concrete, slide number 45 explains it's master index where it contains all words as keys linked to each appearance position within each document; just like this report's implementation.

References

[Dea] Jeff Dean. *Challenges in Building Large-Scale Information Retrieval Systems.* (accessed: 20.09.2019).

[Pra] Selva Prabhakaran. *Cosine Similarity.* URL: `https://www.machinelearningplus.com/nlp/cosine-similarity`. (accessed: 19.09.2019).

[Wik] Wikipedia. *tf–idf.* URL: `https://en.wikipedia.org/wiki/Tf%5C-idf`. (accessed: 19.09.2019).