NUI Galway
OÉ Gaillimh

School of Computer Science

Information Retrieval Assignment 1 - CT5153

Oisin Brannock
20235671

October 19, 2021

An assignment submitted in partial fulfilment
of the requirements for the degree of
MSc. Artificial Intelligence

Declaration

I hereby declare that this assignment is entirely my own work

I have read and I understand the plagiarism provisions, found at https://www.nuigalway.ie/plagiarism/.

Signed: [illegible] Date: 19/10/21

Q1: Document ranking

We have been given 3 documents and 1 query to determine the ranking of the documents with respect to the query.

For this solution, I will use the basic TF-IDF weighting scheme.

- **D1**: Shipment of gold damaged in a fire
- **D2**: Delivery of silver arrived in a silver truck
- **D3**: Shipment of gold arrived in a truck
- **Q**: gold silver truck

From week 2 lectures, we have:

$$w_i j = f_{i,j} log \frac{N}{n_j} = (TF)(IDF)$$

Where $f_{i,j}$ is some function of the frequency of t_i in document d_j, N is the number of documents in the collection and n_i is the number of documents in the collection that contain term t_i.

Here $N = 3$ as we have a corpus comprised of 3 documents.

Therefore, I can code up a solution that will rank the documents [1]. However to illustrate an understanding, the theory of the solution will be discussed.

First what we simply want to know is how often does a term occur in the document. We can then determine the relevance of all terms in a document using IDF. We can multiply these two results together to get TF-IDF. We will have a vector for each document and query that can be used to get the cosine similarity between them and rank the documents.

We have 11 distinct terms across the corpus, so I will demonstrate the calculation for a single term to show how it should go for the other 10.

$$\begin{aligned}
TF_{silver,D1} &= f_{silver,D1} = 0 \\
TF_{silver,D2} &= f_{silver,D2} = 2 \\
TF_{silver,D3} &= f_{silver,D3} = 0 \\
TF_{silver,Q} &= f_{silver,Q} = 1 \\
IDF_{silver} &= log \frac{3}{1} \approx 0.48 \\
TF - IDF_{silver,D1} &= 0 * 0.48 = 0 \\
TF - IDF_{silver,D1} &= 2 * 0.48 = 0.96 \\
TF - IDF_{silver,D1} &= 0 * 0.48 = 0 \\
TF - IDF_{silver,D1} &= 1 * 0.48 = 0
\end{aligned}$$

If we calculate this for each term across each document, we can make a table of the results, or simply store them in vectors. In my code for example, the vector for Q looks like this:

$$\overrightarrow{Q} = \{0, 0.48, 0, 0, 0, 0.18, 0, 0, 0.18, 0, 0\}$$

Now to get the similarity between these document vectors and the query vectors, we can use the following formula from week 2's lecture notes:

$$sim(D_j, Q) = \frac{\overrightarrow{D_j} \cdot \overrightarrow{Q}}{|\overrightarrow{D_j}||\overrightarrow{Q}|}$$

So for example if we want $sim(D_1, Q)$, we use the vectors and get the dot product, and then divide this by the length of each vector multiplied together. The result gives us our ranking score for the document with respect to the query.

$$sim(D_1, Q) = 0.08$$
$$sim(D_2, Q) = 0.82$$
$$sim(D_3, Q) = 0.33$$

Therefore, our document ranking using simple TF-IDF is D_2, D_3, D_1.

If we use the normalised TF-IDF variant, we would see the score of D_1 and D_3 are much closer to D_2. So while the ranking would not change, we would see that this is much more valuable in a scenario where we had a bigger corpus with many more terms, and the difference could be closer and matter.

Q2: Document ranking change based on new terms

Given a situation now where we add terms to our sentences to see if they have an affect on our similarity scores.

D_1 is now "Shipment of gold damaged in a fire. Fire."

The changes would depend on whether the calculations account for capitalisation. Assuming all words are converted to lower case, $sim(Q, D_1)$ would decrease. If this was not the case, then it would still decrease as we are adding an extra term that is not in the query vector. We are simply adding more terms, and as a result, the similarity will decrease as the number of terms increases. This is further compounded if the term is then repeated but to a lesser extent. The first two decrease further if all terms are lower case as we then have 3 "fire." terms showing up. The second two do not because the word "gold" was not already present in D_1.

The results are as follows for illustration of the points above:

- The score if we have D_1 as normal as well as normal case: **0.08**
- The score if we have D_1 as extra Fire using lowercase: **0.05**
- The score if we have D_1 as extra Fire using normal: **0.07**
- The score if we have D_1 as 2 extra Fire using lowercase: **0.04**
- The score if we have D_1 as 2 extra Fire using normal: **0.05**
- The score if we have D_1 as extra Gold using lowercase: **0.07**
- The score if we have D_1 as extra Gold using normal: **0.07**
- The score if we have D_1 as 2 extra Gold using lowercase: **0.05**

- The score if we have D_1 as 2 extra Gold using normal: **0.05**

So to summarise:

- D_1 is now "Shipment of gold damaged in a fire. Fire." - Decreases similarity score.
- D_1 is now "Shipment of gold damaged in a fire. Fire. Fire." - Further decreases similarity score.
- D_1 is now "Shipment of gold damaged in a fire. Gold." - Decreases similarity score.
- D_1 is now "Shipment of gold damaged in a fire. Gold. Gold" - Further decreases similarity score.

Q3: Document ranking of scientific papers

New features we could include in our document collection:

- Citations from papers
- Authors of the papers
- Abstract extraction
- Title extraction

We could incorporate each of these into the manual or machine learning approach of labelling documents as relevant or non relevant to our search. With abstracts, we can take them and have a much smaller sample collection to use with cosine similarity. For something as vast as ACM, this would reduce the query time dramatically, while only reducing the result set quality slightly.

For these kinds of documents, we need to be mindful of things like conversion. Most would be written in LaTeX, which has a lot of formatting and as a result may add a lot of random terms to our documents that decrease query similarity scores when in reality they may be very relevant. So on top of what I have raised above, we need to account for this in terms of removal of stop words, lemmatisation, formatting to a common file structure by getting rid of tags etc. and finally conversion of all words to lowercase.

For the ideas I have raised above, we could use normalised TF-IDF to get good results using cosine similarity. The citations could be connected to each document they come from. As a result we could do something like $sim(Q, C_1)$ to get the similarity of our query to the citation. This could then link back to the paper if the score were high enough to reach a defined threshold. This could also be done for the author, the title or the abstract of the paper.

Bibliography

[1] O. Brannock, "Tf-idf code," (2021).