Assignment 7 COL 106

Arnav Tandon 2022CS11085, Hitesh Yadav 2022MT11322,
Sanya Sachan 2022MT11286, Yash Bansal 2022CS51133

15 November 2023

1 Introduction : Scoring Scheme Issues

In the first part of the assignment, we used the scoring scheme:

$$score(w) = \frac{frequency_1(w) + 1}{frequency_2(w) + 1}$$

.

$frequency_1(w)$: in the Mahatma Gandhi corpus
$frequency_2(w)$: in a general corpus

for any keyword w in the query.

The problem with this scoring scheme is that words which are uncommon in a general corpus but appear repeatedly in the corpus we are using, like "Mahatma" and "Gandhi", will have a very high score. Because of this, very long paragraphs which have the words "Mahatma" and "Gandhi" appearing many times would be scored very high for a query like "What were Mahatma Gandhi's views on the Partition?" even though they may contain nothing about his views on the Partition and are not relevant to the input query.

The expression for IDF decreases with respect to the parameter n_{qi} (logarithm with $base > 1$ is increasing, and the expression in the log is decreasing with respect to n_{qi}). Therefore, for a keyword that appears in many paragraphs, the IDF will be lower, reducing its contribution to the score. Additionally, the value of parameter b can be adjusted to give a higher score to a paragraph with a lower word count. This compensates for the fact that some paragraphs may have a larger size and were receiving a higher score in the Part 1 algorithm primarily because of the greater frequency of unimportant keywords.

Due to these features, we chose the BM25 algorithm for finding the top k paragraphs for the second part of the assignment.

2 BM 25 Algorithm

2.1 Jaccard Coefficient:

A commonly used measure of overlap between two sets A and B is the Jaccard similarity, denoted as $J(A, B)$, defined by the formula:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard similarity always yields values between 0 and 1. For the same set, $J(A, A)$ is always equal to 1.

2.1.1 Issues with Jaccard coefficient :

Doesn't consider the term frequency. Rare terms in a collection are more informative than frequent terms. This fact is overlooked by Jaccard coefficients.

Hence clearly this jaccard coefficient is of little importance in our assignment and hence is mentioned just for theoretical importance.

2.2 Bag of Words Model:

This model does not consider the ordering of words in a document.

2.2.1 Term Frequency(tf)

In part 1 of the assignment, we utilized the term frequency (tf) to determine the number of times a term t occurs in a document d, denoted as $tf_{t,d}$.

2.2.2 Log-frequency weighting

$\text{wt}_d = 1 + \log(t_{ft,d}) \qquad t_{ft,d} > 0$
$wt_d = 0 \qquad t_{ft,d} = 0$

Score = $\sum_{t \in q \cap d}(1 + \log(t_{ft,d}))$

Score is for a document query pair. Sums over terms t in both query and document(paragraph in our interest).

2.2.3 Document Frequency

Rare terms are more informative than others and this takes care of that .Consider a term in the query that is rare in the collection and we want a high weight for rare terms. BM 25 method has to take care of this argument and hence a parameter must be present to decrease the score as the popularity of the word across all the documents increases.

2.2.4 Idf weight

d_{ft} = document frequency of t, i.e., number of documents that contain t.
$d_{ft} \leq N$
$id_{ft} = \log_{10}\left(\frac{N}{d_{ft}}\right)$

We used log instead of direct proportionality to dampen the effect of idf

Idf is designed to *penalize* terms that are very common across all the documents, and at the same time provide perks to terms that are rare or unique.

2.2.5 tf-idf weighting

The tf-idf weight of a term is the product of its term frequency weight and idf weight.

$wt_d = (1 + \log(t_{ft,d})) \cdot (\log_{10}(N/d_{ft}))$

The wt_d increases with the number of occurrences within a document and also increases with the rarity of terms in the collection. Now, wt_d is starting to resemble the formula we have used for the BM25 method.

2.2.6 Documents/Queries as vectors

We have a $|V|$-dimensional vector space, and terms (keywords in the query, but let's assume that it's all the words in the query) are the axes of the space. These are very sparsely spaced vectors since dimensions are practically very high, hence most entries are zero.

The key idea, according to the theory, is to rank documents according to their proximity to the query in this space. We rank documents that are "close" to the query higher.

How to define "proximity"? We can't simply call it the distance between two vectors as that can be arbitrarily large. The angle between two documents is small, denoting similarity. The proximity in angular terms is captured by the cosine function. If the cosine is high, it means that the two vectors have similar angles.

BM25 is an advanced version of what has been discussed above.

2.3 BM 25:

$$Score(D,Q) = \sum_{i=1}^{n} IDF_{q_i} \cdot \frac{f_{q_i,D} \cdot (k_1+1)}{f_{q_i,D} + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}$$

where $\text{IDF}_{q_i} = \log\left(\frac{N - n_{q_i} + 0.5}{n_{q_i} + 0.5}\right)$

.

$\text{f}_{q_i,D}$: term frequency of q_i in document D
k_1 : parameter
b : parameter
$|D|$: length of document D
$avgdl$: average document length in the collection

Here, $f_{q_i,D}$ is the number of times that q_i occurs in the document D, $|D|$ is the length of document D in words, and $avgdl$ is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters.

Here N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

This algorithm is based on a generative model for documents where words are drawn *independently* from the vocabulary using a multinomial distribution. The distribution of term frequencies (tf) follows a binomial distribution – approximated by a Poisson.

2.3.1 Parameters k_1 and b :

k_1 controls term frequency scaling. For high values of k_1, increments in tf_i continue to contribute significantly to the score. Contributions tail off quickly for low values of k_1.

b controls document length normalization. When b is close to 0, the length normalization has less effect, and documents of varying lengths are treated more equally. Taking b closer to 1 gives higher weight to shorter documents.

References

[1] Wikipedia page on BM25

[2] Stanford University Handouts for BM25