CS6370 Assignment 2

Purvam Jain (EE20B101), Deepankar Mishra (CS22M041)

March 2023

1 Assuming are, and, not as stop words, arrive at an inverted index representation for the above documents (treat each sentence as a separate document)

Documents after stopword removal:
S1 Herbivores typically plant eaters meat eaters
S2 Carnivores typically meat eaters plant eaters
S3 Deers eat grass leaves
Inverted Indexing:
This table indexes terms/words to documents based on the frequency of term in the specific document.

	Counts,tf		
Terms	S1	S2	S3
Herbivores	1	0	0
typically	1	1	0
plant	1	1	0
eaters	2	2	0
meat	1	1	0
Carnivores	0	1	0
Deers	0	0	1
eat	0	0	1
grass	0	0	1
leaves	0	0	1

Table 1: Inverted Index table

2 Next, we must proceed on to finding a representation for the text documents. In the class, we saw about the TF-IDF measure. What would be the TF-IDF vector representations for the documents in the above table? State the formula used.

		Counts,tf				weights,w			
Terms	Q	S1	S2	S3	IDF_i	Q	S1	S2	S3
Herbivores	0	1	0	0	0.4771	0	0.4771	0	0
typically	0	1	1	0	0.1761	0	0.1761	0.1761	0
plant	1	1	1	0	0.1761	0.1761	0.1761	0.1761	0
eaters	1	2	2	0	0.1761	0.1761	0.3522	0.3522	0
meat	0	1	1	0	0.1761	0	0.1761	0.1761	0
Carnivores	0	0	1	0	0.4771	0	0	0.4771	0
Deers	0	0	0	1	0.4771	0	0	0	0.4771
eat	0	0	0	1	0.4771	0	0	0	0.4771
grass	0	0	0	1	0.4771	0	0	0	0.4771
leaves	0	0	0	1	0.4771	0	0	0	0.4771

Table 2: Term Vector Model based on $w_i = tf_i * IDF_i$

In the above table we use the Salton's Vector space model where,

$$TermWeight : w_i = tf_i * IDF_i$$

where, tf_i : term frequency in document

$$IDF_i = log(\frac{D}{df_i})$$

where, D : number of documents in database
df_i : number of documents containing the term.
Therefore documents can be represented in term space as:
$S1 = 0.4771 * t1 + 0.1761 * t2 + 0.1761 * t3 + 0.3522 * t4 + 0.1761 * t5$
$S2 = 0.4771 * t6 + 0.1761 * t2 + 0.1761 * t3 + 0.3522 * t4 + 0.1761 * t5$
$S3 = 0.4771 * (t7 + t8 + t9 + t10)$

Here, ti refers to terms in order as in above table.

3 Suppose the query is "plant eaters", which documents would be retrieved based on the inverted index constructed before?

Our query contains two terms "plants" and "eaters", thus documents with non-zero frequency value for both the terms should be retrieved. Looking at the Inverted Index table we can see the documents **S1 and S2** will be retrieved.

4 Find the cosine similarity between the query and each of the retrieved documents. Rank them in descending order.

We know that cosine similarity is given by: $sim(Q, D_i) = \frac{\sum_i (w_{Q,j} w_{i,j})}{|Q||D_i|}$
$|S1| = \sqrt{0.4771^2 + 3 * 0.1761^2 + 0.3552^2} = 0.6684$
$|S2| = \sqrt{0.4771^2 + 3 * 0.1761^2 + 0.3552^2} = 0.6684$
$|S3| = \sqrt{4 * 0.4771^2} = 0.9542$
$|Q| = \sqrt{2 * 0.1761^2} = 0.2489$
$Q \cdot S1 = 0.1761^2 + 0.1761 * 0.3522 = 0.0929$
$Q \cdot S2 = 0.1761^2 + 0.1761 * 0.3522 = 0.0929$
$Q \cdot S3 = 0$
$Cosine\theta_{S1} = \frac{Q \cdot S1}{|Q| * |S1|} = 0.5599$
$Cosine\theta_{S2} = \frac{Q \cdot S2}{|Q| * |S2|} = 0.5599$
$Cosine\theta_{S3} = \frac{Q \cdot S3}{|Q| * |S3|} = 0$
Therefore, S1 and S2 will be ranked together as Rank 1 and S3 will be ranked as Rank 2.

5 Is the ranking given above the best?

No, ideal ranking should be $S1 > S3 > S2$, since the word "grass" is close to "plant" and word "eater" is close to "eat". Semantically, looking at the query, we should rank documents on plant eaters or herbivores higher than documents on carnivores.

6 Vector Space Model

6.1 What is the IDF of a term that occurs in every document

IDF of a term occurring in every document will be 0.

6.2 Is the IDF of a term always finite? If not, how can the formula for IDF be modified to make it finite?

No, IDF can become infinite if the denominator is 0 , that is the term doesn't exist in the document corpus. For this reason we can employ Laplace smoothing as follows:
$IDF = log(\frac{N+V}{n+1})$
Here, N = total number of documents, n = frequency , V = total words in the corpus. By adding 1, we smooth/transfer the probability of occurrence of the term.

7 Can you think of any other similarity/distance measure that can be used to compare vectors other than cosine similarity. Justify why it is a better or worse choice than cosine similarity for IR.

Other popular measures of distance/similarity are as follows:

- **Euclidean Distance**
  - Calculated as $L_2 - norm$ between the two vectors.
  - Documents have more terms compared to queries and hence the distance between them will always be large.
  - Cosine similarity is scale invariant and hence better compared to this measure.
- **Manhattan Distance**
  - Calculated as $L_1 - norm$ between the two vectors
  - Suffers from the same disadvantage of scale imbalance in queries and documents and hence cosine similarity is better
- **Jaccard Similarity**
  - It measures the similarity between two vectors by computing the ratio of the size of their intersection(terms in common dimension) to the size of their union(terms in all dimensions in space).
  - We only reply on term frequency here with no regards to IDF, which is a much better measure.
- **Pearson correlation coefficient**
  - Measures the linear correlation between two vectors -1 implying negative correlation, 0 means no correlation and 1 means positive correlation

– Only good at capturing linear relations and thus fails to perform better than cosine similarity.

Cosine similarity is best suited for IR tasks, because it is invariant to scale and length, as well can capture similarity between both sparse and dense vectors.

8 Why is accuracy not used as a metric to evaluate information retrieval systems?

$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$
Accuracy fails to handle class imbalance well. In IR tasks, most of the time, the number of True negatives will be close to the total volume of the corpus. Accuracy is insensitive to True positives or the number of relevant documents since they are really small. If an IR system is evaluated on accuracy, it can always give values close to 1 by deeming all documents as True negatives or non-relevant, which is bad performance. Precision and Recall are better measures of effectiveness for IR systems.

9 For what values of α does the F_α -measure give more weightage to recall than to precision?

$F_\alpha = \frac{1}{\frac{\alpha}{precision}+\frac{1-\alpha}{recall}}$

- $\alpha \approx 1$, $F_\alpha = precision$
- $\alpha \approx 0$, $F_\alpha = recall$
- For $\alpha \in [0, 0.5)$ more weight to recall
- For $\alpha \in (0.5, 1]$ more weight to precision
- For $\alpha = 0.5$, same weight to both

10 For what values of $alpha$ does the F$alpha$ -measure give more weightage to recall than to precision?

For $\alpha \in [0, 0.5)$ more weight to recall

11 What is a shortcoming of the Precision @ k metric that is addressed by Average Precision @ k?

Precision fails to capture the order in which items are recommended. Average Precision @k considers the order

12 What is Mean Average Precision (MAP) @ k? How is it different from Average Precision (AP) @ k?

Mean Average Precision(MAP) for a set of queries is the mean of the average precision scores for each query. For calculating MAP we need to calculate AP@k for each query, then take their averages(divide by number of queries).

13 For the Cranfield dataset, which of the following two evaluation measures is more appropriate and why? (a) AP (b) nDCG

In the cranfield dataset we have different levels of relevance of a document for a given query, 1 indicates exact match to the query, 4 indicates not relavant. Hence nDCG is a better evaluation measure because it rates IR performance better if a highly relevant document is retrieved in a lower rank (as in CLose to 1). AP can be good if relevance is binary, that is, either a document is relevant or it is not.

14 Implement the following evaluation metrics for the IR system: a. Precision @ k b. Recall @ k c. F-Score @ k d. Average Precision @ k e. nDCG @ k

Please refer to the code.

15 Assume that for a given query, the set of relevant documents is as listed in cranqrels.json. Any document with a relevance score of 1 to 4 is considered as relevant. For each query in the Cranfield dataset, find the Precision, Recall, F-score, Average Precision and nDCG scores for k = 1 to 10. Average each measure over all queries and plot it as function of k. Code for plotting is part of the given template. You are expected to use the same. Report the graph with your observations based on it.

Recall increases as the rank at which we observe it increases. Precision falls down, and the harmonic mean, which is Fscore, is somewhere between precision and recall. nDCG increases as the rank at which we observe it increases, although the performance of this IR model is poor when it comes to fetching documents related to semantically related queries.

Please see the graph from the following link. It can also be found on running main.py. Graph of Evaluation Metrics

16 Analyze the results of your search engine. Are there some queries for which the search engine's performance is not as expected? Report your observations

The precision value decreases from around 0.4 at the start, which is average performance. The model does not work well for custom queries where due to synonymous words the simmilarity score gets decreased for matching doc and Ids. Recall grows as we check for higher number of Retrieved results. COnsider a query which is "assemble replicas of aerodynamically non rigid", which is

17 Do you find any shortcoming(s) in using a Vector Space Model for IR? If yes, report them.

1. Long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality)

2. Documents with similar context but different vocabulary won't be matched, resulting in false negative matches.

3. The order in which the terms appear in the document is lost in the vector space representation.

18 While working with the Cranfield dataset, we ignored the titles of the documents. But, titles can sometimes be extremely informative in information retrieval, sometimes even more than the body. State a way to include the title while representing the document as a vector. What if we want to weigh the contribution of the title three times that of the document?

If a term in query is found in the title of a doc, then we should increase the similarity of doc and that particular query.

1. Compute the TFIDF weights of each term in the title and body separately.

2. Multiply the TFIDF weight of each term in the title by a real number greater than 1 and the TF-IDF weight of each term in the body by 1.

3. Add the weighted title and the weighted body vectors to create the final document vector.

4. Now calculate similarity between query and Doc.

19 Suppose we use bigrams instead of unigrams to index the documents, what would be its advantage(s) and/or disadvantage(s)?

Advantages of using bigrams instead of unigrams to index documents-

Now we can capture context too in the similarity calculation. Reduces ambiguity by capturing phrases/expressions that have a specific meaning. Bigrams give better recall, thus more relevant documents are retrieved.

Disadvantages of using bigrams instead of unigrams to index documents-

Now the computation cost in terms of time will be very high compared to the unigrams implementation. Increases sparsity, thus making it more difficult to find relevant documents. Increases noise by considering irrelevant and bigger terms.

20 In the Cranfield dataset, we have relevance judgements given by the domain experts. In the absence of such relevance judgements, can you think of a way in which we can get relevance feedback from the user himself/herself? Ideally, we would like to keep the feedback process to be non-intrusive to the user. Hence, think of an 'implicit' way of recording feedback from the users.

One way to implicity recording feedback is to record the amount of time a user spends on a particular document, and the number of time he/she opens the document. We can assign a value of time-spend * number of clicks on the document to a particular document as it relevance score.

21 References

- Lecture Notes and Slides
- Similarity and Distance measures

Find complete code here : Code Repository