Assignment 02
REPORT
Team: 41

CS19M007: Abhinav Anurag
NA18B030: Ramakrishnan
CS21M069: Vardhman Anand Uikey

1. **Now that the Cranfield documents are pre-processed, our search engine needs a data structure to facilitate the 'matching' process of a query to its relevant documents. Let's work out a simple example.**
   Consider the following three sentences:
   S1 Herbivores are typically plant eaters and not meat eaters
   S2 Carnivores are typically meat eaters and not plant eaters
   S3 Deers eat grass and leaves

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

Answer:

An inverted index is a data structure that stores a mapping between information, such as words or integers, and their places in a document or group of documents. So, to calculate the inverted index representations the data needs to be pre-processed. The code for pre-processing is as follows:-

Output:

```
['Herbivores', 'typically', 'plant', 'eater', 'meat', 'eater']
['Carnivores', 'typically', 'meat', 'eater', 'plant', 'eater']
['Deers', 'eat', 'grass', 'leaf']
```

Source Code:
https://colab.research.google.com/drive/1b69IziY7OP40ToW3cak6Ym5xWj8Sdihb#scrollTo=aNHRbv4S9DGS

The table of phrase frequencies for three pre-processed sentences while treating each sentence as a separate document is shown below: -

Words	S1	S2	S3	Inverted Index Representation
Herbivores	1	0	0	$\begin{pmatrix}1\\0\\0\end{pmatrix}$
typically	1	1	0	$\begin{pmatrix}1\\1\\0\end{pmatrix}$
plant	1	1	0	$\begin{pmatrix}1\\1\\0\end{pmatrix}$
eater	2	2	0	$\begin{pmatrix}1\\1\\0\end{pmatrix}$
meat	1	1	0	$\begin{pmatrix}1\\1\\0\end{pmatrix}$
Carnivores	0	1	0	$\begin{pmatrix}0\\1\\0\end{pmatrix}$
Deers	0	0	1	$\begin{pmatrix}0\\0\\1\end{pmatrix}$
eat	0	0	1	$\begin{pmatrix}0\\0\\1\end{pmatrix}$
grass	0	0	1	$\begin{pmatrix}0\\0\\1\end{pmatrix}$
leaf	0	0	1	$\begin{pmatrix}0\\0\\1\end{pmatrix}$

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.

Answer:

The term frequency-inverse document frequency (TF-IDF) is a statistical metric that determines how relevant a word is to a document in a set of documents.
This is accomplished by multiplying two metrics: Term Frequency and the word's inverse document frequency over a group of documents in the corpus. Documents S1, S2, and S3 have the following term frequencies:

Doc1 =

1
1
1
2
1
0
0
0
0
0

Doc2 =

0
1
1
2
1
1
0
0
0
0

Doc3 =

0
0
0
0
0
0
1
1
1
1

I D F =

Term Frequency

Term frequency (TF) is the number of times a term appears in a document divided by the number of words. It can be denoted in the vector representation as a vector having the dimensions equal to the number of words and each entry in the vector corresponds to the number of occurrences of that word in the particular document.

*

Inverse Document Frequency

This indicates how frequent or uncommon a term is over the full text collection. The closer it is to zero, the more prevalent the term. This metric may be computed by taking the total number of documents, dividing them by the number of documents that include a word, and then computing the logarithm.

IDF = $\log(d/D)$

d = number of documents in which the word occurs.

D = Total number of Documents in the corpus.

TF - IDF = *TF of the document * IDF of corresponding word.*

IDF Values

Words	IDF
Herbivores	**0.47**
typically	**0.17**
plant	**0.17**
eater	**0.17**
meat	**0.17**
Carnivores	**0.47**
Deers	**0.47**
eat	**0.47**
grass	**0.47**
leaf	**0.47**

TF-IDF Values

	Doc 1	Doc 2	Doc 3
Herbivores	0.47 * 1 = 0.47	0.47 * 0 = 0	0.47 * 0 = 0
typically	0.17 * 1 = 0.17	0.17 * 1 = 0.17	0.17 * 0 = 0
plant	0.17 * 1 = 0.17	0.17 * 1 = 0.17	0.17 * 0 = 0
eater	0.17 * 2 = 0.34	0.17 * 2 = 0.34	0.17 * 0 = 0
meat	0.17 * 1 = 0.17	0.17 * 1 = 0.17	0.17 * 0 = 0
Carnivores	0.47 * 0 = 0	0.47 * 1 = 0.47	0.47 * 0 = 0
Deers	0.47 * 0 = 0	0.47 * 0 = 0	0.47 * 1 = 0
eat	0.47 * 0 = 0	0.47 * 0 = 0	0.47 * 1 = 0.47
grass	0.47 * 0 = 0	0.47 * 0 = 0	0.47 * 1 = 0.47
leaf	0.47 * 0 = 0	0.47 * 0 = 0	0.47 * 1 = 0.47

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

Answer:

The documents with non-zero TF-IDF value will be retrieved. Consequently,

The query ‘plant’ retrieves Document S1 and Document S2.
The query ‘eaters’ retrieves Document S1 and Document S2.

So, the query “plant eaters” would retrieve the union of the documents retrieved in first and second query. Therefore, Document 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.

Answer:

The query is *"plant eaters"* and the retrieved documents are Document S1 and Document S2.
Cosine Similarity Calculation

$|Doc1| = sqrt(0.47^2 + 0.17^2 + 0.17^2 + 0.34^2 + 0.17^2) = sqrt(0.4232) = 0.6505$

$|Doc2| = sqrt(0.17^2 + 0.17^2 + 0.34^2 + 0.17^2 + 0.47^2) = sqrt(0.4232) = 0.6505$

$|Doc3| = sqrt(0.47^2 + 0.47^2 + 0.47^2) = sqrt(0.6627) = 0.8140$

$Query = |\text{'plant eaters'}| = sqrt(0.17^2 + 0.17^2) = sqrt(0.0578) = 0.2404$

Dot Products

Doc 1 = 0.17*0.17 + 0.17*0.35 = 0.0884
Doc 2 = 0.17*0.17 + 0.17*0.35 = 0.0884
Doc 3 = 0

CosineDoc1 = 0.0884 / 0.6505 = 0.13
CosineDoc2 = 0.0884 / 0.6505 = 0.13
CosineDoc3 = 0 / 0.2404 = 0

The document S1 and S2 have the same cosine similarity for the given query and Document S3 has the least rank.

S1 > S2 > S3 (Descending order of Ranking)

5. Is the ranking given above the best?

Answer:

The ranking of document S1 and document S2 are equal. "Deers eat grass and leaves" has similar meaning to "plant eaters" but Document S3 has the least ranking. Semantically, S2 should have the least ranking. Accordingly, this ranking is not best if we evaluate syntactically. S1 should have the highest ranking and then S3.

6. Now, you are set to build a real-world retrieval system. Implement an Information Retrieval System for the Cranfield Dataset using the Vector Space Model.

Answer:

Implementation provided in the file “informationRetrieval.py”

7.
(a) What is the IDF of a term that occurs in every document?

Answer:

IDF = log $\frac{D}{d}$

D = Total number of documents in the corpus.
d = Number of documents in which the word occurs

if a term occurs in every document then d = D.
So, log(D/d) = 0
So, the IDF is equal to zero in this case.

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

Answer:

If d = 0 that is if the word does not occur in any document, then the IDF will become infinite as log(D/0) is considered as infinite.

Modification required

This can be modified so that it is always finite by saying that the frequency of each term is always greater than or equal to one and the number of documents under consideration is always limited and finite.

8. 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.

Answer:

We can use the following similarity/distance measure other than the cosine similarity for comparing vectors:

a) Jaccard similarity
b) Euclidean distance
c) Manhattan distance

While analysing text similarity, Jaccard similarity is useful when duplication is not an issue, but cosine similarity is useful when duplication is an issue. It is preferable to utilise Jaccard similarity for two product descriptions since repetition of a term does not lessen their similarity. Jaccard similarity only considers a single set of words for each phrase or text, whereas cosine similarity considers the overall length of the vectors. (These vectors might be created using a bag of words, term frequency, or tf-idf.)

Jaccard similarity over union is the size of intersection divided by size of union of two sets.

$J(A, B) = |A \cap B| / |A \cup B| = |A \cap B| / |A| + |B| - |A \cap B|$

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

Answer:

Accuracy of an Information Retrieval System is defined as:

$$\text{Accuracy} = \frac{True\ Positive + False\ Positive}{True\ Positive + False\ Positive \ + \ False\ Negative + True\ Negative} = \frac{A+D}{A+B+C+D}$$

such that:

	Relevant documents	Not Relevant documents
Retrieved documents	A	B
Not Retrieved documents	C	D

Accuracy is not used as a statistic to evaluate information retrieval systems since, in virtually all cases, the data is severely skewed: typically, more than 99.9% of the documents fall into the non-relevant group.
A, B, and C are so tiny in comparison to D, the accuracy value for all IR systems will be near to one. In other words, because the accuracy is not highly sensitive to A, B, and C, comparing different accuracies becomes extremely difficult.

A system configured for maximum accuracy might appear to perform well by simply ignoring all documents that are irrelevant to all queries. Even if the system is fairly effective, labelling certain documents as relevant will almost always result in a significant proportion of false positives. Labelling all papers as irrelevant, on the other hand, is entirely disappointing to an information retrieval system user.

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

Answer:

Fα = (1 + α^2) *Precision * Recall / (α^2*Precision + Recall)

where:
α is such that Recall is considered α times as important as Precision.

So, any value of α greater than 1 gives more importance to recall than precision.

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

Answer:

Precision@k = Number of relevant results in top k retrieved results.

Shortcoming: Rank of the retrieved documents are not considered while calculating precision.

However in case of average precision @ k this shortcoming is addressed.

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

Mean Average Precision (MAP) @ k is a metric to evaluate the average precision across multiple queries. It can is the mean of average precision for all the queries.

MAP = $1/Q \Sigma_{q=1 \text{ to } Q} AP(q)$

Where
Q = Total number of Queries.
AP(q) = Average precision for query q.

The average precision is calculated for a single query. However, in order to assess an IR system, we must analyse it for a large number of questions. The mean average precision (MAP@k) is calculated by taking the mean of the Average Precision values (AP@k) for Q queries.

As a result, this is a superior IR assessment approach to AP@k.

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

Answer:

nDCG

Explanation:

nDCG considers graded relevance, whereas AP considers merely relevance, i.e. whether the document is relevant or not. Because the Cranfield dataset includes previous relevance judgments by domain experts, we can utilise it to construct nDCG, which provides a superior assessment metric for our IR system than AP, which does not take this crucial component of the dataset into account.

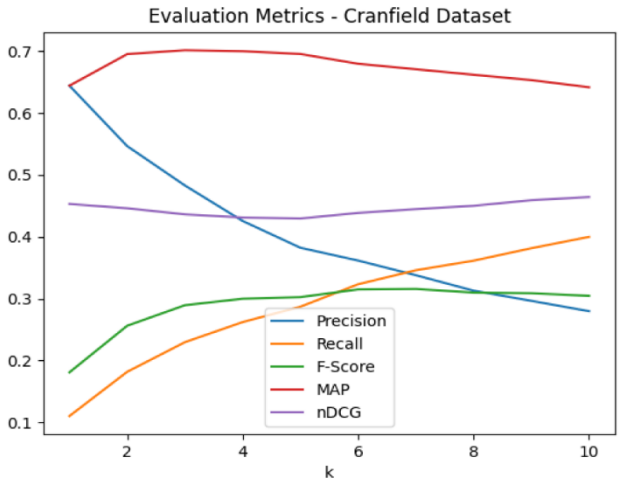
14. Implement the following evaluation metrics for the IR system: (a) Precision @ k (b) Recall @ k (c) F-Score @ k (d) Average Precision @ k (e) nDCG @ k

Answer:

The above evaluation metrics have been implemented for the IR System. Implementation provided in the file “Evaluation.py”.

15. Assume that for a given query, the set of relevant documents is as listed in *cran_qrels.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**.

Answer:



Observations

a) The value of Precision reduces with k.
b) The value of Recall increases with k.
c) The recall and MAP values at k=1 are the same
d) F1-score is very low.
e) The nDCG value does not fluctuate much and remains pretty steady for all values of k.

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

Answer:

a) For varying choices of k, the F-score and nDCG values for this IR system are insufficient. This is due to the usual constraints of utilising a vector space model.
b) Because the whole IR system is built on matching words, the system will fail if the titles and their associated documents do not share enough words. In the Cranfield dataset, there are occasions when the title has no terms in common with the most relevant text.
c) Because of constraints of lemmatization, the presence of immediately before or succeeding punctuation (without spaces left from the previous or subsequent words) might skew the findings.
d) Because all of the documents are about aerodynamics, the number of words they share is quite vast, and therefore the number of discriminating terms is relatively small, making it impossible to pick the most relevant materials.

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

Answer:

The shortcomings of the vector space model are as follows:

a) The Vector Space Model may fail to recognize synonyms, which means that papers with identical context but distinct word vocabulary will not be coupled.
b) The order of the words in the texts is not taken into account in the vector space model's unigram representation.
c) The vector space model involves computations using high-dimensional vectors and is thus computationally intensive.
d) Poor inner product with documents having similar meanings but different vocabularies.
e) Failure to remove polysemous documents

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?

Answer:

We can consider the words of the titles to be included in the vocabulary in order to incorporate titles in the vector model. The tf-idf score of the title words can be multiplied by three to give them three times the importance of the body words. Each document in the corpus may be thought of as consisting of the body and the title three times, i.e. three repeats of the title combined with the words of the body. In this case, the title's phrases would be assigned three times more weight, and our IR system would perform far better.

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

Answer:

Advantages

- Bigrams solve the major drawback of unigrams, which is that they do not capture the sequence in which the words appear.
- Bigrams utilise context information more effectively and hence produce more precise results than unigrams.

Disadvantages

- Using bigrams is substantially costlier computationally than using unigrams and greatly increases the dimensionality of the vectors.

- Another downside of employing bigrams is that they have a poorer recall than unigrams, which means that it may not recover some texts that have comparable terms but no matching context or surrounding words. This creates an issue in the case of IR inquiries, because most questions are supplied with simply pertinent phrases poured in without properly spelling out the necessary surrounding words or worrying about syntax.

- The vector space model's originality assumption is considerably less likely to hold in the case of bigrams since two bigrams with the same word are evidently not independent of each other.

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.

Answer:

When a specific query is searched for, one implicit way of recording feedback from users is to record the documents opened by the users and the time spent on them. For example, if a user searches for the query "Q" and the returned documents are a collection of ids with values such as id1, id2, id3, id4, and if the user views id1 and spends 5 seconds on it before opening id2 and spending 1 minute on it, id2 will be awarded a higher relevance score than id1. And, depending on the results of several users behaviour pattern, such scores may be averaged to provide a greater relevance score for each of the documents. Thus, measuring the time spent on a document by each user when looking for a certain query may be used as a metric to derive relevant assessments.

When a user enters a query and the results are presented, the result that the user clicks on is more likely to be relevant than the other papers. As a consequence, data from numerous searches and users may be used to enhance results.

References

I. https://monkeylearn.com/blog/what-is-tf-idf/
II. https://nlp.stanford.edu/IR-book/html/htmledition/queries-as-vectors-1.html
III. https://towardsdatascience.com/calculate-similarity-the-most-relevant-metrics-in-a-nutshell-9a43564f533e
IV. https://en.wikipedia.org/wiki/Vector_space_model
V. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52