CS6370 (NLP): Assignment 2

Manikandan Sritharan (EE19B038) & Snehan J (EE19B027)

1 Inverted Index Representation for the given documents

Inverted index is a representation for each word that shows what all documents a given word is present in. Ignoring the stop words, we can efficiently represent it as a dictionary, whose keys are words, while values are lists containing document IDs.

Words	Documents
Herbivores	S1
Typically	S1, S2
Plant	S1, S2
Eaters	S1, S2
Meat	S1, S2
Carnivores	S2
Deers	S3
Eat	S3
Grass	S3
Leaves	S3

2 TF-IDF measure for the given documents

Here, in this measure, we use a vector representation of words. We assume the terms to be orthogonal and represent the document as a linear combination of the terms.

$$\text{Weight (of a particular term to a document)} = localweight \times globalweight$$

$$\text{Weight} = TF \times IDF$$

$$\text{TF (term frequency)} = \text{Number of occurences of a word in a document}$$

$$\text{IDF (Inverted Document Frequency)} = \frac{N}{n}$$

N is the total number of documents and n is the number of documents containing the given word. The table showing the Term frequency is as follows:

Words	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

The table showing the IDF values is as follows:

Words	IDF
Herbivores	$\log 3$
Typically	$\log 1.5$
Plant	$\log 1.5$
Eaters	$\log 1.5$
Meat	$\log 1.5$
Carnivores	$\log 3$
Deers	$\log 3$
Eat	$\log 3$
Grass	$\log 3$
Leaves	$\log 3$

The table showing the final weights of words to a document in the vector space of interest is shown below:

Words	S1	S2	S3
Herbivores	0.477	0	0
Typically	0.176	0.176	0
Plant	0.176	0.176	0
Eaters	0.352	0.352	0
Meat	0.176	0.176	0
Carnivores	0	0.477	0
Deers	0	0	0.477
Eat	0	0	0.477
Grass	0	0	0.477
Leaves	0	0	0.477

The TF-IDF **vector representation of a document is its column in the above table**, as it shows the weights of words corresponding to each document.

3 Documents retrieved for a given query using Inverted Index

The given query is "plant eaters". The queries "plant" and "eaters" both retrieve documents S1 and S2.

The document S3 will not be returned as it does not contain either of the words. Hence S1 and S2 will be retrieved.

4 Ranking documents using cosine similarity

The given query is "plant eaters".

We consider the query itself to be a document and we find the TF-IDF value (the correct linear combination of words) of the given query.

Words	TF	IDF	weight
Herbivores	0	0.477	0
Typically	0	0.176	0
Plant	1	0.176	0.176
Eaters	1	0.176	0.176
Meat	0	0.176	0
Carnivores	0	0.477	0
Deers	0	0.477	0
Eat	0	0.477	0
Grass	0	0.477	0
Leaves	0	0.477	0

The cosine similarity between a query and given document is defined as:

$$\text{cosine similarity score } (\overrightarrow{doc},\overrightarrow{query}) = \frac{\overrightarrow{doc}\cdot\overrightarrow{query}}{|\overrightarrow{doc}|\times|\overrightarrow{query}|}$$

Applying the above score formula, we find the scores of the three documents with respect to the query.
Score of document S1 and query = $\frac{0.093}{0.248\times0.666} = 0.563$
Score of document S2 and query = $\frac{0.093}{0.248\times0.666} = 0.563$
Score of document S3 and query = 0
Greater the cosine similarity score, better the rank of the document. Thus, documents S1 and S2 are retrieved and are ranked the same by this method.

5 Is the ranking given above the best?

According to cosine similarity, we assign the same rank to documents S1 and S2. But, we would want to rank S1 higher than S2 because of context. S1 talks about herbivores which are actually "plant eaters", but S2 talks about carnivores, which is logically less related.

6 Building an IR system using vector space model

It has been implemented in the informationRetrieval.py code.

7 Inverted Document Frequency

7.1 a) IDF of a term that occurs in every document

$$\text{IDF (Inverted Document Frequency)} = \log\frac{N}{n}$$

N is the total number of documents and n is the number of documents containing the given word. If a term occurs in every document, n = N and hence

$$\text{IDF} = \log 1 = 0$$

IDF of a term that occurs in every document is 0.

7.2 b) Changes made to IDF to make it finite

Now, when a term does not appear in any of the documents i.e n = 0

$$\text{IDF} = \log\frac{N}{0} = \infty$$

Thus, we add one to the denominator. Now, when N = n, we have $\log \frac{N}{N+1}$, which is negative.
So, we add one to the numerator as well.
Finally, we have

$$\text{IDF} = \log \frac{N+1}{n+1}$$

8 Other similarity measures

Jaccard Similarity: Defined as Intersection-over-Union of the words present in the query and document. It measures what fraction of words are common between them. The Jaccard similarity index does not consider repeated occurrences of words. Meanwhile, the union of words continues to increase for larger documents, while the intersection remains small due to the small query size. Hence, Jaccard index does accentuate the differences in word content between doc and query well enough.
Minkowski Distance:
For p = 1 and 2, it is known as Manhattan and Euclidean distance respectively. Without normalising with vector lengths, Euclidean distance takes into account the frequency of words in the document. This can be problematic if the dataset contains documents with different sizes, as large documents will have long vectors, while queries will mostly be short.

9 Why is accuracy not used as a metric in IR systems?

Accuracy isn't a great measure for IR systems because of **class imbalance**.

$$Accuracy = \frac{TruePositives+TrueNegatives}{Totalno.ofdocuments}$$

True positives would mean the relevant documents that are retrieved and true negatives would mean the irrelevant documents that are not retrieved.
Usually, the True negative value is supposed to be large in an IR system (due to the large no. of documents). Therefore, the accuracy is almost one, and its being insensitive to the true negatives and the false positives and false negatives. Hence, accuracy isn't a good metric for IR systems.

10 F_α measure

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

The above formula applies for $\alpha \in [0,1]$.
For $\alpha \in [0,0.5)$, more weightage is given to recall.
For $\alpha = 0.5$, equal weightage is given.
For $\alpha \in (0.5,1]$, more weightage is given to precision.
Therefore, for $\alpha \in [0,0.5)$, this measure gives more weightage to recall than precision.

11 Shortcoming of Precision @k addressed by Average Precision (AP) @k

$$\text{Precision @k} = \frac{\text{Number of relevant documents in the k retrieved documents}}{k}$$

Assuming the number of relevant documents within rank k remains the same, Precision @ k is invariant of the ranking of the relevant documents, which is not ideal.

$$\text{Average Precision @k} = \frac{\sum precision@i * rel(i)}{\text{Number of relevant documents}}$$

Since $Precision@i$ generally decreases with i, $AveragePrecision@k$ would be higher if the relevant documents are ranked higher.

12 Mean Average Precision (MAP) @k

The Average precision @k is specific to a query, so we get different values for different queries.
The Mean Average Precision @k takes a mean of Average Precision @k over the queries. This can be used to compare different systems.

$$\text{Average Precision @k} = \frac{\sum Averageprecision@k(\text{over all queries})}{\text{Total no. of queries}}$$

13 AP vs nDCG

The Average precision @k measures the precision by taking the documents as either relevant or not relevant (no percentage relevance). Also, it does not tell if one document is more relevant than the other.

$$\text{DCG} = \sum \frac{relevance_i}{\log_2(i+1)}$$

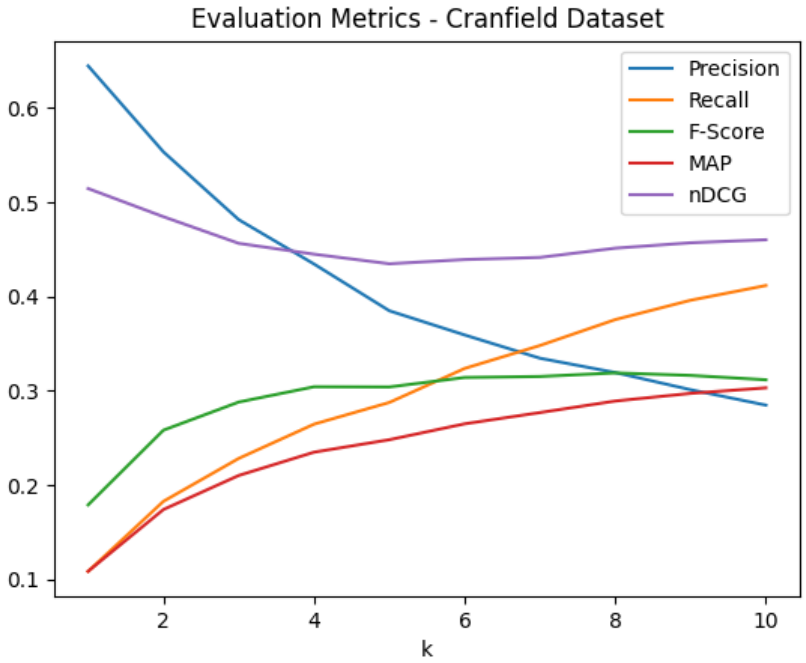
$$\text{nDCG} = \frac{DCG}{iDCG}$$

iDCG is the DCG for ideal ranking of documents.
The nDCG metric considers the above and evaluates the score while considering the relevance ranking as a criteria using the hierarchy of relevance and hence performs better.

14 Implementing evaluation metrics

It has been implemented in the evaluation.py code.

15 Plot and Observations



15.1 Observations:

- **Precision** decreases with k. This can be understood by walking through the rankings of the retrieved documents. At the start, the search engine will retrieve the documents which are the most similar to the query and hence have high precision. However, with the increase in k, the engine will move to less confident documents and hence the precision decreases with k.
- **Recall** is a non-decreasing function of k. This is because the denominator (number of relevant documents in the corpora) is the same, and the numerator (number of relevant documents retrieved) can only increase or stay constant with respect to k.
- **F-score** which represents a trade-off between Precision and Recall saturates after $k \geq 4$. In a setting where you give equal importance to precision and recall, the model's performance saturates after $k \geq 4$.
- **MAP** increases with k. This is because the average precision is computed by adding precision values at all relevant documents and dividing it by the number of relevant documents in the corpora (which is a constant).
- **nDCG** is approximately the same over all k, which implies that the model is performing good over all k.

16 Analysis of results of the search engine

- Since the search engine is based on the Vector Space Model, it inherently cannot model word senses. This can result in a few unexpected performance on some queries.
- Consider a query which has no words in common with any of the documents but the words are synonyms of existing words in the corpora. In this case, our model will randomly return documents, which is of course, not ideal.
- Another case where our search engine under performs is when we have general queries like "sonic". We would like to see results of documents containing "supersonic", "subsonic", "hypersonic", etc. However since the model relies on stemming, it will not be able to mine these relations and hence these documents will not show up in the results.

17 Shortcomings in the Vector Space Model for IR

As a basic model, the term vector scheme discussed has several limitations.

- First, it is very calculation intensive. From the computational standpoint it is very slow, requiring a lot of processing time.
- Second, each time we add a new term into the term space we need to recalculate all vectors.
- The order in which the terms appear in the document is lost in the vector space representation.
- Long Documents: Very long documents make similarity measures difficult (vectors with small dot products and high dimensionality).
- False negative matches: documents with similar content but different vocabularies may result in a poor inner product. This is a limitation of keyword-driven IR systems.
- Semantic content: Systems for handling semantic content may need to use special tags (containers).

18 Using the title of a document in IR

We used TF-IDF measure in our vector space to calculate the similarity measure score. We essentially calculated the weights for each term in a document.
For the Cranfield dataset, we find that the document sizes aren't long, so we could actually include the title as a sentence in the document. The weight for the term in the title could be thrice the weight calculated by the TF-IDF representation (owing to the importance of titles in document retrieval).

In reality (other datasets), we would have to multiply the weight of the terms in the title by a large number owing to the size of the large size of the documents.

19 Bigrams vs Unigrams

Advantages of using bigrams over unigrams:

- Word order or sequence (context) is captured by bigrams.
- Bigrams have more precision in document retrieval.

Disadvantages:

- Computationally expensive. The vector space is 26×26 D, instead of 26 D in the case of unigrams.
- Increased time taken for retrieval of documents.
- We assume orthogonality in vector space model and this is even less likely to be true in the case of bigrams than for unigrams. Also, bigrams with a common letter will be dependant on each other.
- Reduced recall as it may not retrieve documents with similar words in the query.

20 Relevance feedback from the user

Instead of asking the users directly for feedback, we can rather record how the user behaves when the IR system gives the result and analyse the behavior.
The relevance of a document can be found out based on the number of times the user has clicked on a given document. User's recent search history can also be used to give relevant documents.
The amount of time the user spends on the document can also be used for relevance. Documents having higher viewtime can be preferred.

21 References

- https://en.wikipedia.org/wiki/Jaccard_index
- https://en.wikipedia.org/wiki/Minkowski_distance