CS6370: Natural Language Processing

Team details:

Team number: **16**
Team members:

1) N Kausik (CS21M037)
2) Karthikeyan S (CS21M028)
3) Kavya Rengaraj (CS21Z018)

Assignment 2

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:

herbivores	**S1**
typically	**S1,S2**
plant	**S1,S2**
eaters	**S1,S2**
meat	**S1,S2**

carnivores	**S2**
deers	**S3**
eat	**S3**
grass	**S3**
leaves	**S3**

Code Snippet for inverted index:

```
docs = ["Herbivores are typically plant eaters and not meat eaters", "Carni
vores are typically meat eaters and not plant eaters", "Deers eat grass and
leaves"]
stop_words = ["are","not","and"]
updated_docs = []
for doc in docs:
    # convert to lower case
    doc = doc.lower()
    # remove stop words
    for stop_word in stop_words:
        doc = doc.replace(stop_word, "")
        # remove whitespace inbetween words
    doc = " ".join(doc.split())
    updated_docs.append(doc)
# create inverted index of updated documents
inverted_index = {}
for doc in updated_docs:
    words = doc.split()
    for word in words:
        if word not in inverted_index:
            inverted_index[word] = []
        if 'S'+str(updated_docs.index(doc)+1) not in inverted_index[word]:
            inverted_index[word].append('S'+str(updated_docs.index(doc)+1))
print(inverted_index)
```

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:

Formula for TF-IDF :

TF-IDF = TF*IDF

TF(word)= no. of occurrences of a term in the document ‘d’ / no. of terms in the document ‘d’

IDF = log(no. of documents / no.of documents the term occurs)

	herbivores	**typically**	**plant**	**eaters**	**meat**	**carnivores**	**deers**	**eat**	**grass**	**leaves**
S1	0.445047207612279 5	0.338469872682 6178	0.338 46987 26826 178	0.676 93974 53652 356	0.3384 69872 68261 78	0.0	0.0	0.0	0.0	0.0
S2	0.0	0.338469872682 6178	0.152 71512 19790 2584	0.676 93974 53652 356	0.3384 69872 68261 78	0.445047 2076122 795	0.0	0.0	0.0	0.0
S3	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.5	0.5

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

Answer:

The retrieved documents are

S1 Herbivores are typically plant eaters and not meat eaters

S2 Carnivores are typically meat eaters and not plant eaters

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

Answer:

Documents	Cosine similarity score
S1	0.71800303
S2	0.71800303
S3	0.0

5. Is the ranking given above the best?

Answer:

No, the ranking given is not the best because of the following reasons:

1. Only direct matching of terms in the query with terms in the document is done. This doesn't capture the semantic information in the documents
2. For the query "plant eaters" both s1 and s2 are given the same rank even though they are semantically different (S1 - plant eaters, S2 - not plant eaters).
3. Since we removed stop words during preprocessing, we are not able to distinguish between **'plant eaters'** and **'not plant eaters'**. So, generally we can't distinguish between contradicting words.

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: Refer **informationRetrieval.py** in the submitted zip file.

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

Answer:

The IDF of a term that occurs in every document is 0 (i.e. Total no.of documents will be equal to the number of documents the term occurs)

IDF = log(no. of documents/ no.of documents the term occurs) = log(1/1)=0

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

Answer:

No. While querying, when a term in a query that is not present in any of the documents occurs, its document frequency will be 0.

log(1/0)= Infinite.

To make it finite, 1 is added to the IDF.

Modified Formula:

TF(word) = no. of occurrences of a termin the document 'd' / no. of terms in the document 'd'

IDF = log(no. of documents / (no.of documents the term occurs + **1**))

TF-IDF = TF*IDF

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:

Euclidean Distance: It is a **worse** choice since we are working with vectors, and we care more about the similarity in the directions of the vectors than their magnitudes, euclidean distance is not a good measure since it calculates distance between vectors including magnitude and direction. In applications relating to probabilities, this cannot be used since its value is not bounded between 0 to 1. However Euclidean distance can be used in cases where magnitude is also important.

Jaccard Similarity: Since Jaccard similarity is calculated as intersection over union of the set of unique values in the vector, it will perform better in cases where the vectors are like sentences with words and **repetition** does not affect the output.

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

Answer:

Accuracy is a bad metric for information retrieval because in most cases, for any query most of the documents are irrelevant. This is due to the extreme skewness observed in the data.

Accuracy = (True Positives + True Negatives) / (TP+TN+FN+FP)

In real life data, the #Positives (relevant documents) for a query are very low and hence the True Negatives will dominate over the True Positives in the accuracy formula. Hence, even if the system retrieves many non relevant documents (high False Positive, low True Positive), the accuracy will still be high which makes it a bad metric for this use case. Hence we should not use accuracy for information retrieval systems.

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

Answer:

$$F = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}} = \frac{1}{\frac{1}{P_{eff}} + \frac{1}{R_{eff}}}$$

From the formula, we can infer that,

By increasing $\propto$ from 0 to 1, ($0 <= \propto <= 1$)

- $\propto$/P increases => P/$\propto$ decreases => P_eff decreases
- Hence weightage of P decreases
- (1-$\propto$) decreases => (1-$\propto$)/R decreases => R/(1-$\propto$) increases => R_eff increases
- Hence weightage of R increases

At $\propto$ = 0.5, equal weightage is given.

Hence for more weightage to recall than precision, **0.5 < $\propto$ < 1.0**.

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

Answer:

Shortcoming is that Precision @ k does not account for the position of the relevant items in the ordered retrieved documents when calculating the value.

Eg.

> Consider for a model A, **first 2** retrieved documents out of **6** are relevant according to ground truth.
>
> For model B, **the last 2** retrieved documents out of **6** are relevant according to ground truth.
>
> Here, Precision @ 6 = 2/6 for both A and B.
>
> But, Average Precision @ 6 for A will be greater than for B.
>
> Clearly since the first 2 retrieved documents are expected to be more relevant to the query, A is supposed to be a better model than B since it gave the most relevant documents for the query correctly.

This issue is addressed by Average Precision @ k since here we consider the positions of the correctly retrieved documents also in calculation.

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

Answer:

Mean Average Precision @ k is an evaluation metric which can be computed simply as taking the mean of Average Precision @ k for many queries.

$$MAP\ @\ k\ =\ \frac{\sum_{q=1}^{Q} AveragePrecision(q,k)}{Q}$$

In information retrieval, AP @ k is generally used for evaluation over a single query but MAP @ k is used as a single evaluation metric over all queries. I.e. AP @ k gives a value for every query but MAP @ k gives only a single value for all queries.

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

Answer:

nDCG is more appropriate for Cranfield dataset since we are provided with the positions in "qrels" which can be used to find relevance scores which are used in nDCG. Since AP does not consider this score, nDCG captures more information than AP and hence it is a better metric.

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

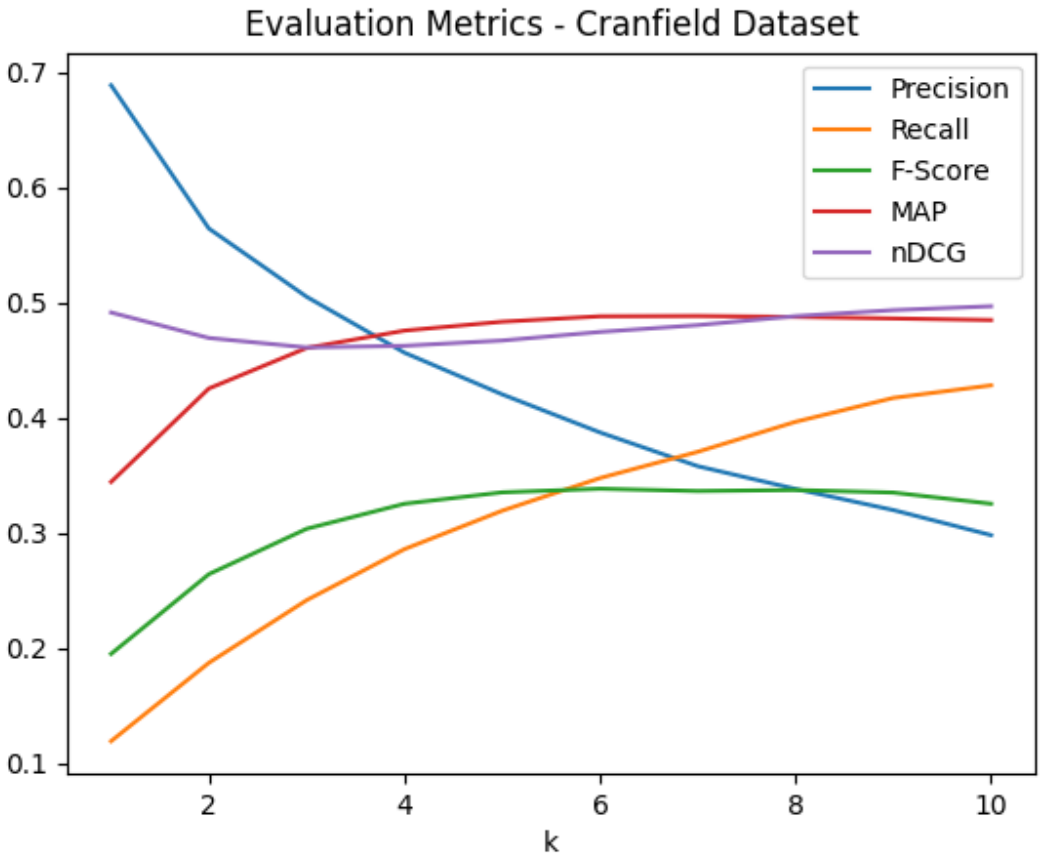
Answer: Refer **evaluation.py** in the submitted zip file.

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 a 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:

Graph:

Segmenter: Punkt
Tokeniser: PTB



Observations:

Behaviors:

1. Precision: Decreases with increase in k
2. Recall: Increases with increase in k
3. F-Score: Increases but saturates after k = 5 at around 0.335
4. MAP: Increases but saturates after k = 5 at around 0.488
5. nDCG: Decreases and Increases and ends up at 0.497

Inferences:

1. As k is increased, Precision reduces but all other metrics increase or dont change
2. After k = 5, both MAP and nDCG do not improve much

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:

Query: 19

Does there exist a good basic treatment of the dynamics of re-entry combining consideration of realistic effects with relative simplicity of results .

Actual top 5 relevant documents: 499, 32, 67, 164, 639
Top 5 relevant docs as per our IR: 554, 82, 1346, 660, 1219
Here in Doc 554, there are a lot of common occurrences of words like “re-entry” and hence it is predicted as the most relevant. However the query is about entry of objects into a planet but the document 554 is about heat transfer and they are not related.
The ground truth most related doc is 449 which is semantically related with the query but since there are not many common terms, it isn't predicted.

Query: 28
what application has the linear theory design of curved wings .
Actual top 5 relevant documents: 512, 279, 224
Top 5 relevant docs as per our IR: 752, 1075, 674, 451, 247
Here in Doc 752, there are a lot of common occurrences of words like “wing”. However in Doc 512 (ground truth) common occurrences are low. However, the query is about curved wings but document 752 is about “slender wing theory” which is semantically different from “curved wings”.
The ground truth most related doc is 512 which is semantically related with the query but since there are not many common terms, it isn't predicted.

Query: 35
Are there any papers dealing with acoustic wave propagation in reacting gases .
Actual top 5 relevant documents: 166, 167, 517, 132
Top 5 relevant docs as per our IR: 1244, 1208, 75, 654, 1327
Here in Doc 1244, there are a lot of common occurrences of words like “acoustic”. However in Doc 166 (ground truth) common occurrences are low. However, the query is about “reacting gases” but document 1244 is about “turbulent jet noise” which is semantically different from “reacting gases”.
The ground truth most related doc is 166 which is semantically related with the query but since there are not many common terms, it isn't predicted.

General Observations:
Polysemy: In many cases, some words are used in different contexts in different documents. But, since the system operates using TF-IDF, it fails to take into account the polysemous nature of these words (bank as in money bank vs bank as in river bank), the IR system fails in such queries which have these words.

Synonymy: In some cases, synonyms are used in the document (acoustic vs sound) and hence the same word is not detected in query and document. Hence the true relevant documents are not retrieved by the IR system.

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

Answer:

Even though the Vector Space Model (VSM) is the most popular model for the Information Retrieval, it has the following limitations:

Using the Vector Space model results in matching of exact words instead of looking into its context. Thus the model remains semantically insensitive.

Synonymy words (different words meaning the same) might be resulting in false negatives.

Vector space model doesn't care about the order in which the words appear in the document and thus the ordering is lost.

Large documents are not represented properly due to limited dimensions in the VSM and hence they are predicted to have poor similarity even though the documents may be similar.

Vector space model theoretically assumes that the terms are statistically independent because VSM assumes that the axes represented by the words are orthogonal. However this may not be the case in real life data.

VSMs cannot deal with lexical ambiguity and variability. For eg. consider the sentences
S1: “My neighbor got ill because of smallpox”
S2: “Variola virus is a deadly virus”
S3: “The server crashed because of a virus”

Even though we know that smallpox is a disease caused by Variola virus, just because S1 and S2 have no common words, in VSM the similarity is zero and they are considered orthogonal. But the sentences S1 and S2 are related.
To the contrary, S2 and S3 have no relation but the similarity between S2 and S3 is not zero as they have the common word “virus”.

VSMs can't be used effectively if we want to model domain relations, as the count of domain specific words will be very less when compared to the non-domain specific words and thus the vectors representing the domain specific words will be very sparse which leads to a near-zero similarity between domain specific words.

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 include the title while representing the document by combining the words that appeared in the title to the words that appeared in the body and we can add a weight factor(k) to the word counts while considering them for TF-IDF.

The new Term occurrences would be

(k * No. of times the word appeared in the title)
+ (No. of times the word appeared in the body)

As stated in the question we can assign k=3,
Hence, Term occurrences = (3 * No. of times the word appeared in the title)
+ No. of times the word appeared in the body

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

Answer:

Advantages:

Indexing the documents using bigram makes more sense compared to indexing them using unigram, as bigram enables us to know the immediate neighbors of the current word and thus carries more information about the context and ordering of the words in the sentences.

Using bigram thus improves the precision of the whole system.

Disadvantages:

Representing the document using bigram increases the computational cost as we have a comparatively increased set of axes (terms) for the representation of documents.

Using bigram over unigram reduces the recall of the system. Thus, if we want more recall than precision then we must use unigram over bigram.

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:

Here we can use the technique of implicit feedback, a method of automatically tracking users' preferences by monitoring the actions they perform such as what are the documents/web links they clicked, the pages they visited, products they purchased, etc,.

For the Cranfield dataset, we can monitor the following measures to arrive at a good feedback process:

- the number of times a document being opened for a particular query(accessed)
- the time they spend on a particular document

Thus, we can give a higher relevance factor to documents opened more than once for a query and also we can give more relevance to the documents on which the users spend more time and vice versa.

i.e, A document accessed many times for a particular query can be given higher relevance and if the average time spent on a document is high, we can give higher relevance.
The feedback can be appropriately obtained by repeating this technique with multiple users on the same query.

References used:

Q9:

https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html

Q20:
https://nlp.stanford.edu/IR-book/html/htmledition/indirect-relevance-feedback-1.html

Q17:
https://en.wikipedia.org/wiki/Vector_space_model#Limitations

http://mlwiki.org/index.php/Vector_Space_Models#Problems