# CS6370 Natural Language Processing

Jan – May 2021

## Assignment 1 – Part 2


ME17B158 - Omkar Nath

ME17B170 – Uma T V

**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).**

First of all, we write the code to pre-process the data and extract the pre-processed sentences.

```
# setup

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

```
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"

stop_words = set(stopwords.words('english'))
f_S1= [lemmatizer.lemmatize(w) for w in word_tokenize(S1) if w not in stop_words]
f_S2= [lemmatizer.lemmatize(w) for w in word_tokenize(S2) if w not in stop_words]
f_S3= [lemmatizer.lemmatize(w) for w in word_tokenize(S3) if w not in stop_words]

print(f_S1)
print(f_S2)
print(f_S3)
```

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

Given below is the table of term frequencies for three pre-processed sentences (documents).

| | **S1** | **S2** | **S3** |
|---|---|---|---|
| **Herbivores** | 1 | 0 | 0 |
| **typically** | 1 | 1 | 0 |
| **plant** | 1 | 1 | 0 |
| **eater** | 2 | 2 | 0 |
| **meat** | 1 | 1 | 0 |

| | | | |
|---|---|---|---|
| **Carnivores** | 0 | 1 | 0 |
| **Deers** | 0 | 0 | 1 |
| **eat** | 0 | 0 | 1 |
| **grass** | 0 | 0 | 1 |
| **leaf** | 0 | 0 | 1 |

The inverted index representation for every non-stop word gives the frequency of the word in every document in corpus. Hence, the inverted index representations are:

Herbivores: $\begin{pmatrix}1\\0\\0\end{pmatrix}$ typically: $\begin{pmatrix}1\\1\\0\end{pmatrix}$ plant: $\begin{pmatrix}1\\1\\0\end{pmatrix}$

eater: $\begin{pmatrix}1\\1\\0\end{pmatrix}$ meat: $\begin{pmatrix}1\\1\\0\end{pmatrix}$ Carnivores: $\begin{pmatrix}0\\1\\0\end{pmatrix}$

Deers: $\begin{pmatrix}0\\0\\1\end{pmatrix}$ eat: $\begin{pmatrix}0\\0\\1\end{pmatrix}$ grass: $\begin{pmatrix}0\\0\\1\end{pmatrix}$

leaf: $\begin{pmatrix}0\\0\\1\end{pmatrix}$

**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.**

The Term Frequency (TF) of each document is a vector with dimension equal to the number of words, and every number as the number of occurrences of that word in the particular document. The Term Frequencies of documents S1, S2 and S3 are:

$$S1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad S2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad S3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

The Inverse Document Frequency (IDF) is calculated for every word in the corpus. It gives an idea about how efficient the word is for differentiating between documents. The formula for calculating the IDF is given by:

$$\text{IDF} = \log\left(\frac{N:\ Total\ number\ of\ documents\ in\ collection}{n\ :number\ of\ documents\ in\ hich\ the\ word\ occurs}\right)$$

Hence, if a word is present in only a few documents, then it's IDF will be high and so will be its discriminative power.

In this case, the IDFs of the non – stopword words of corpus are:

| | | |
|---|---|---|
| Herbivores | log (3) | 0.477 |
| typically | log (1.5) | 0.176 |
| plant | log (1.5) | 0.176 |
| eater | log (1.5) | 0.176 |
| meat | log (1.5) | 0.176 |
| Carnivores | log (3) | 0.477 |
| Deers | log (3) | 0.477 |
| eat | log (3) | 0.477 |
| grass | log (3) | 0.477 |
| leaf | log (3) | 0.477 |

Now, we will compute the TF-IDF Matrix.

TF - IDF vector value for each document = TF of the document * IDF of corresponding word.

TF – IDF Matrix

| | S1 | S2 | S3 |
|---|---|---|---|
| Herbivores | 0.477 * 1 | 0.477 * 0 | 0.477 * 0 |
| typically | 0.176 * 1 | 0.176 * 1 | 0.176 * 0 |
| plant | 0.176 * 1 | 0.176 * 1 | 0.176 * 0 |
| eater | 0.176 * 2 | 0.176 * 2 | 0.176 * 0 |
| meat | 0.176 * 1 | 0.176 * 1 | 0.176 * 0 |
| Carnivores | 0.477 * 0 | 0.477 * 1 | 0.477 * 0 |
| Deers | 0.477 * 0 | 0.477 * 0 | 0.477 * 1 |

| | | | |
|---|---|---|---|
| eat | 0.477 * 0 | 0.477 * 0 | 0.477 * 1 |
| grass | 0.477 * 0 | 0.477 * 0 | 0.477 * 1 |
| leaf | 0.477 * 0 | 0.477 * 0 | 0.477 * 1 |

TF-IDF representations of Documents (in word space)

| | | |
|---|---|---|
| $S1 = \begin{pmatrix} 0.477 \\ 0.176 \\ 0.176 \\ 0.352 \\ 0.176 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ | $S1 = \begin{pmatrix} 0 \\ 0.176 \\ 0.176 \\ 0.352 \\ 0.176 \\ 0.477 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ | $S1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.477 \\ 0.477 \\ 0.477 \\ 0.477 \end{pmatrix}$ |

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

- The documents retrieved from the query ‘plant’ will be Documents S1 and S2 (because they have non – zero TF- IDF value of the word)
- The documents retrieved from the query ‘eaters’ will be Documents S1 and S2.

Hence the documents retrieved from the query ‘plant eaters’ will be a union of the results of the individual words; hence this retrieves documents S1 and S2.

*References:*
https://nlp.stanford.edu/IR-book/html/htmledition/queries-as-vectors-1.html

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

First of all, we consider the query as a small document (bag of words) and find out its TF-IDF representation. The TF – IDF representation of the query ‘plant eaters’ is:

$$q = \begin{pmatrix} 0.477 * 0 \\ 0.176 * 0 \\ 0.176 * 1 \\ 0.176 * 1 \\ 0.176 * 0 \\ 0.477 * 0 \\ 0.477 * 0 \\ 0.477 * 0 \\ 0.477 * 0 \\ 0.477 * 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0.176 \\ 0.176 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Now, we find the cosine similarity between the query and the retried documents (S1 and S2):

$$score(q,d) = \frac{\vec{V}(q).\vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$

$$\text{score(q,S1)} = \frac{\begin{pmatrix}0\\0\\0.176\\0.176\\0\\0\\0\\0\\0\\0\end{pmatrix}.\begin{pmatrix}0.477\\0.176\\0.176\\0.352\\0.176\\0\\0\\0\\0\\0\end{pmatrix}}{\left|\begin{pmatrix}0\\0\\0.176\\0.176\\0\\0\\0\\0\\0\\0\end{pmatrix}\right|\left|\begin{pmatrix}0.477\\0.176\\0.176\\0.352\\0.176\\0\\0\\0\\0\\0\end{pmatrix}\right|} = \frac{0.092928}{0.2489*0.6666} = 0.56$$

$$\text{score(q,S2)} = \frac{\begin{pmatrix}0\\0\\0.176\\0.176\\0\\0\\0\\0\\0\\0\end{pmatrix}.\begin{pmatrix}0\\0.176\\0.176\\0.352\\0.176\\0.477\\0\\0\\0\\0\end{pmatrix}}{\left|\begin{pmatrix}0\\0\\0.176\\0.176\\0\\0\\0\\0\\0\\0\end{pmatrix}\right|\left|\begin{pmatrix}0\\0.176\\0.176\\0.352\\0.176\\0.477\\0\\0\\0\\0\end{pmatrix}\right|} = \frac{0.092928}{0.2489*0.6666} = 0.56$$

Hence, the two retrieved documents, S1 and S2, have same cosine similarity of 0.56 from the given query.

**5. Is the ranking given above the best?**

The ranking above ranks S1 and S2 at **equal** levels. This ranking is **not the best** because logically, the first document being about Herbivores should be **more similar** to the query of ‘plant eaters’ than the second document, which is about Carnivores.

**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.’**

The code has been written and implemented in the attached file “informationRetrieval.py”

**7.**

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

$$\text{IDF} = \log\left(\frac{N:\ Total\ number\ of\ documents\ in\ collection}{n\ :number\ of\ documents\ in\ hich\ the\ word\ occurs}\right)$$

Here, N = n (if the term occurs in every document)

IDF = log(1) = **0**
Thus, the IDF of the term that occurs in every document if 0.

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

We know that:

$$\text{IDF} = \log\left(\frac{N:\ Total\ number\ of\ documents\ in\ collection}{n\ :number\ of\ documents\ in\ hich\ the\ word\ occurs}\right)$$

If the word does not occur in any document then n = 0 and IDF will be infinite.

The formula for IDF can be modified as:

$$\text{IDF} = \log\left(\frac{N+V}{n+1}\right)$$

Where V = total number of words in corpus. This would ensure that:

- The IDF never becomes infinite
- The sum of the probabilities of all words in corpus ( $\sum\frac{n+1}{N+V}$ ) is 1.

**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.**

Some similarity / distance measures that can be used to compare vectors other than cosine similarity are:

<u>Category 1 : Similarity Based Metrics:</u>

<u>Jaccard similarity</u>

- The Jaccard similarity index compares members for two sets to see which members are shared and which are distinct.
- It's a measure of similarity for the two sets of data, with a range from 0% to 100%.
- The higher the percentage, the more similar the two populations.

$$\text{Jaccard Index} = \frac{X \cap Y}{X \cup Y} = \frac{Intersection\ of\ the\ sets\ of\ non{-}zero\ dimensions\ between\ the\ vectors}{Union\ of\ the\ sets\ of\ non{-}zero\ dimensions\ between\ the\ vectors}$$

- The advantage this method has over cosine similarity is that it is simple and easy to interpret.
- However, in spite of this small advantage, this is a worse choice for information retrieval compared to cosine similarity because it is extremely sensitive to small samples sizes. It cannot be used for large documents because the index would be very small for such cases and it would be extremely difficult to compare the relevance of the documents.

Category 2: Distance Based Metrics

Euclidean distance

- The Euclidean distance is the L2 norm between two vectors.

Euclidean distance = $\left|\left|\overrightarrow{V_1} - \overrightarrow{V_2}\right|\right|_2$

- It does not work well in information retrieval, because usually the scales of the queries would be very less compared to the scales of the document. Hence, in spite of being similar, we will obtain high values of distances.

Manhattan distance

- The Manhattan distance is the L1 norm between two vectors.

  Manhattan distance = $\left|\left|\overrightarrow{V_1} - \overrightarrow{V_2}\right|\right|_1$

- This also does not work well in information retrieval, because of the same reasons as for the Euclidean distance, that the scale differences between queries and documents will distort the results.

*References:*

https://towardsdatascience.com/calculate-similarity-the-most-relevant-metrics-in-a-nutshell-9a43564f533e

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

Accuracy of an Information Retrieval System is defined as:

Accuracy = $\frac{A+D}{A+B+C+D} = \frac{\text{True Positives + True Negatives}}{Total\ number\ of\ Documents}$

Where:

| | Relevant documents | Not Relevant documents |
|---|---|---|
| Retrived documents | A | B |
| Not Retrived documents | C | D |

Accuracy is not used as a measure to evaluate information retrieval systems because D (true negatives) is usually very large, because of the large size of corpus. Hence, since A, B and C are very small compared to D, the accuracy value will be close to 1 for all IR systems. In other words, the accuracy is not very sensitive to A, B and C and hence, it becomes really difficult to compare different accuracies.

**10. For what values of α does the $F_\alpha$-measure give more weightage to recall than to precision?**

The $F_\alpha$- measure gives the weighted Harmonic mean of Precision and Recall. It is used when we want to evaluate IR systems with custom weightages given to precision and recall.

$F_\alpha = \frac{1}{\frac{\alpha}{Precision}+\frac{1-\alpha}{Recall}}$ ; $\alpha \in [0,1]$

α gives the amount of weightage given to precision.

| **α** | **$F_\alpha$** |
|---|---|
| 0 | Recall |
| Between 0 and 0.5 | More weightage to Recall |
| Between 0.5 and 1 | More weightage to Precision |
| 1 | Precision |

Hence, α gives more weightage to Precision when it is in (0.5, 1].

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

Precision@k (P@k) is defined as:

P@k = $Number\ of\ relevant\ results\ in\ top\ k\ retrived\ results$

However, this measure does not take into account the ranking of relevant document retrieval. For example, an IR system having first two relevant results and last two relevant results out of 10 will both have a P@k value of 2, in spite of the first IR system clearly doing a better work than the second. This shortcoming of P@k is addressed by the Average Precision @ k (AP@k).

AP@k = $\frac{\sum_{i=1}^{k} P@i * rel(i)}{Number\ of\ Relevant\ Documents}$

Where rel(i) = 0 if document I is not relevant and 1 if it is relevant

Here, we take the average of the P@i values for all i <=k where document i is relevant. Clearly, this measure takes ranking into account. In the previous example, if the first two out of 10 documents were relevant, then the AP@10 for that would be 2/10 = 0.2, on the other hand if the last two out of 10 were relevant, it would be (1/9+1/10)/10 = 0.021, which is lower than the first, as intended.

*References:*
https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

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

The average precision is query specific, which means that it is taken for a specific query. However, inorder to evaluate an IR system, it is required that we evaluate them for many queries. The mean average precision (MAP@k) takes the mean of the Average Precision values (AP@k) for Q number of queries.

$$MAP@k = \frac{\sum_{q=1}^{Q} AP@k_q}{Q}$$

Hence, this is a better IR evaluation method than AP@k.

*References:*
https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

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

As we have seen earlier, the Average Precision gives us a measure of the average of the precisions at different ranks. It gives us a good estimate of how good an IR system is in retrieving the relevant documents at higher ranks. However, one major assumption of rather a simplification factor in this evaluation measure is that it is assumed that the documents are either relevant or not relevant. There is nothing in between, Also, there is no notion of on document being more relevant than the other being used in this evaluation method. This is therefore a major limitation as n the real world, out of all relevant documents, we can actually make a hierarchy of which is more relevant and which is comparatively less relevant. The nDCG evaluation method overcomes this drawback.

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

where DCG (Discounted Cumulative Gain) = $\sum_{i=1}^{n} \frac{relevance_i}{\log_2 i+1}$
and iDCG is the DCG for the ideal ranking of documents.

The nDCG method uses the relevance number of every document (indicative of the hierarchy of relevance) and uses that to calculate the effectiveness of the IR system, while also taking into account ranking.

Hence, nDCG is more appropriate as it has all the features of the AP method and it also overcomes the limitation of AP method and takes into account the order of relevance of documents.

*References:*
https://blog.thedigitalgroup.com/measuring-search-relevance-using-ndcg

**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**

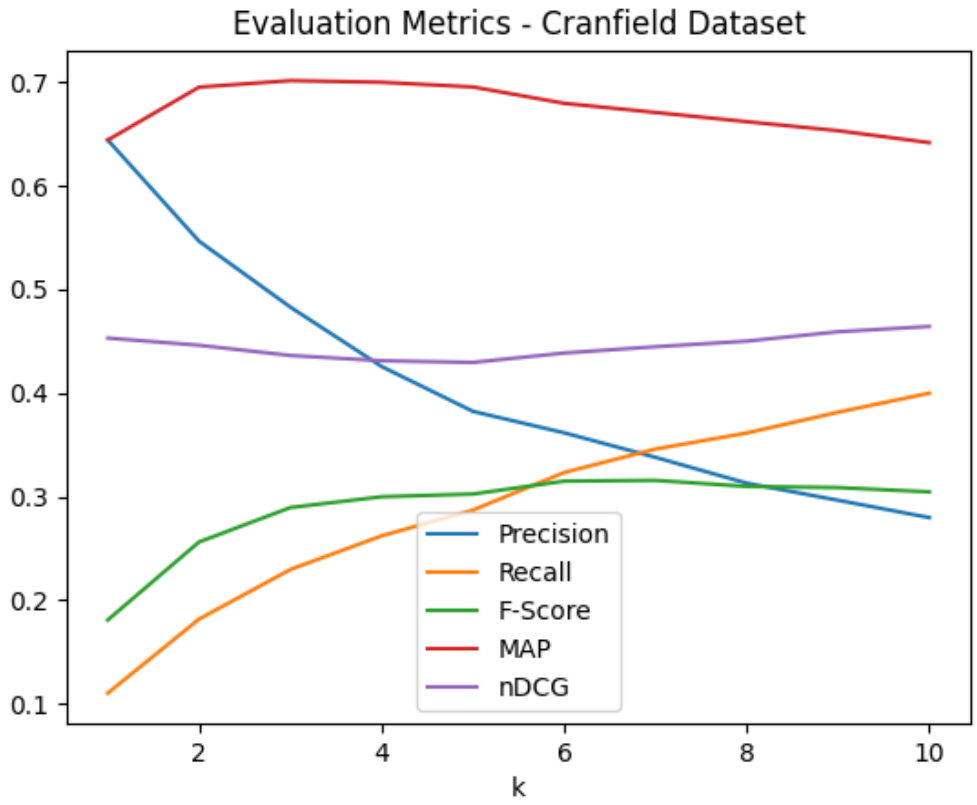
The code for the evaluation metrics has been implemented in the attached file "evaluation.py".

**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.**

The Precision, Recall and F-Score values obtained on the Cranfield dataset for different values of k are as in the table below.

| K | Precision | Recall | F-Score | MAP | nDCG |
|---|---|---|---|---|---|
| 1 | 0.6444444444444445 | 0.10991668579586084 | 0.18063320256997412 | 0.6444444444444445 | 0.4530370370370746 |
| 2 | 0.5466666666666666 | 0.18147007781911884 | 0.25599961157179246 | 0.6955555555555556 | 0.4460857362755255 |
| 3 | 0.4829629629629304 | 0.22940939368528784 | 0.28924505426338065 | 0.7018518518518852 | 0.4362331533115904 |
| 4 | 0.4255555555555556 | 0.2619126096539534 | 0.29970232159292837 | 0.7002469135802469 | 0.4311369067130723 7 |
| 5 | 0.3822222222222225 | 0.2868095394260122 | 0.30231224737333223 | 0.6957283950617281 | 0.4294905267027 9875 |
| 6 | 0.3614814814814135 | 0.3231278471293775 | 0.31477989416543817 | 0.6799160493827716 | 0.4385930340659391 |
| 7 | 0.3377777777777779 | 0.3458095366712536 | 0.31551425359536833 | 0.6710691358024469 | 0.4446874906539 0256 |
| 8 | 0.3127777777777777 | 0.3612845419129253 | 0.30967084387141183 | 0.6621573948097755 | 0.4500812872370 42 |
| 9 | 0.2962962962962966 | 0.3814141071456 8963 | 0.30855588624193026 | 0.6534542579994963 | 0.4591500487483 8794 |
| 10 | 0.2795555555555558 | 0.3998093292075 7845 | 0.30440593014620193 | 0.6419187032837826 | 0.4642605830057 3786 |

The graph plotting the results obtained is as shown below:




OBSERVATIONS:

- The value of Precision reduces with k. This is because: there are two cases, one if the added document is relevant, in which case the resultant precision will be:
$$\frac{(number\ of\ relevant\ document\ till\ k-1)+1}{(k-1)+1} < \frac{number\ of\ relevant\ document\ till\ k-1}{k-1}$$
(since it is a proper fraction) , and the other case when the added doc is not relevant, in which case the precision again decreases:
$$\frac{(number\ of\ relevant\ document\ till\ k-1)}{(k-1)+1} < \frac{number\ of\ relevant\ document\ till\ k-1}{k-1}$$
Hence, precision always decreases with k.
- The value of Recall increases with k. This is because when a new document id added it is either relevant, in which case the recall increases, or is not relevant, in which case the recall remains constant.
- The F-score first increases, reaches a maximum value at k = 8 and then decreases. This implies that the Information Retrieval System with top 8 extracted documents is the best for Information Retrieval.
- The recall and MAP values at k=1 are the same. This can also be theoretically verified with the formulae.
- The nDCG value doesn't observe lot of fluctuations and is fairly constant for all values of k, which implies that the IR system works well for all first k values, when the ranked relevance of the documents is taken into account.

**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.**

- Since the entire IR system works based on matching words, the system won't work when the titles and their relevant documents don't have enough words in common. There are instances in the Cranfield dataset wherein the title has no common words with its most relevant document!
- Since all the documents are about aerodynamics, the number of words the documents have in common is very large and hence the number of discriminatory words is very less, which makes it difficult to extract the most relevant documents.
- The F-score and nDCG values for this IR system are not good enough for different values of k. This is because of the general limitations of using a vector space model for information retrieval system, which are described in the next question.
- Sometimes, the presence of immediately preceding or succeeding punctuations (without spaces left from the neighbouring words) distort the results due to limitations from the lemmatization end.

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

The following are some shortcomings of the vector space model:

- Practically, since there are many words in the corpus, and since documents are also usually long, the similarity values can be poorly represented as they will be small in magnitude in spite of having a good number of matching words.
- This model is based of the orthogonality assumption, according to which different words are orthogonal to each other. This becomes a huge limitation for synonyms which are different words with the same context, but this similarity will not be captured at all in the vector space model.
- In unigram representation in the vector space model, the order in which the words occur in the documents is not considered at all.
- The vector space model has calculations involving high dimensional vectors and hence is computationally expensive.
- The vector space model is not modular, in the sense that the addition of a new word in the model would require changing all the vectors, which is computationally expensive.

*References:*
https://en.wikipedia.org/wiki/Vector_space_model

**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?**

For extracting documents in our Information Retrieval System, we use the TF-IDF (Term Frequency- Inverted Document Frequency) of the documents to measure similarity. The TF-IDF representation gives information about the weighted frequency of every word in the document (term frequency multiplied by the discriminative power of the word) If we want to add the title information also while measuring similarity for IR, one obvious way would be to just consider the title as a line of the document and recalculate the TF-IDF representation, including the title. If we want to weigh the contribution of the title three times that of the document, then we can just include three repetitions of the sentence in the document while calculating the TF-IDF.

One point to consider here is that documents in the real life are normally very large. In such cases, just adding a few words of the title may not significantly change the similarity results, or in other words, the information provided by the title may get lost in the information provided by the huge documents. In such cases, we will have to increase the weight of the title even more to make it comparable to the influence of the document words. Luckily, in the Cranfield dataset, the documents are not very large and we can include the title as a part of the document with a repetition of 3 to include the information provided by it properly.

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

The advantages and disadvantages of using a bigram instead of unigram to index documents are:

| Advantages | Disadvantages |
|---|---|
| • The main limitation of unigrams, of not capturing the order in which the words occur, is overcome by using bigrams.<br>• Bigrams use the context information better and hence have higher precision of results compared to unigrams | • Using bigrams is computationally much more expensive compared to using unigrams and increases the dimensionality of the vectors by a huge factor.<br>• The originality assumption of the vector space model is even less likely to hold in case of bigrams because in case of two bigrams with a common word, they are obviously not independent of each other.<br>• Another disadvantage of using bigrams is that it has a lower recall compared to unigrams, that is, it may not retrieve some documents with similar words but without matching context or surrounding words. This becomes a problem in case of IR with queries, wherein most queries are given with just relevant words being dumped in without actually typing out the correct surrounding words and without caring out the grammar. |

**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.**

Everything the user does on the information retrieval system can be recorded and used to improve the results in an implicit way.

- The history of the user can be used to resolve the issues of polysemy, wherein the results can be shown for that meaning / context that has been searched for earlier. For example, if the user gives the query 'banks' in the search engine, the IR system can show results for financial banks if the user had search for investment options earlier and the river bank if the user had searched for nature earlier.
- Once the user gives the query and the results have been displayed, the result that the user clicks on is more likely to be more relevant compared to the remaining documents. Hence, this information from multiple searches and users can be used to improve results.
- The documents with maximum visits and time spent by users can be given more weightage on relevance compared to the less often visited documents and documents that have less average view time. Furthermore, the documents that recorded more activity from the users can also considered more relevant (for example, in case of web searches, login, purchase, feedback given to the website, forms filled in the website etc. can all be used in information retrieval as well.
- The information about the area of the screen or the part of the results the user had his/ her curser on and the space of the screen the user was looking at can be all be used to improve results.