CS6370: Natural Language Processing
Project

Release Date: 21st March 2024 Deadline: 15/05/2024

Name:	Roll No.:
Shivaram V	BE20B032
Boneshwar VK	BS20B012
Aakash B	BS20B001

General Instructions:

1. The template for the code (in Python) is provided in a separate zip file. You are expected to fill in the template wherever instructed. Note that any Python library, such as nltk, stanfordcorenlp, spacy, etc, can be used.
2. A folder named 'Roll_number.zip' that contains a zip of the code folder and your responses to the questions (a PDF of this document with the solutions written in the text boxes) must be uploaded on Moodle by the deadline.
3. Any submissions made after the deadline will not be graded.
4. Answer the theoretical questions concisely. All the codes should contain proper comments.
5. For questions involving coding components, paste a screenshot of the code.
6. The institute's academic code of conduct will be strictly enforced.

The first assignment in the NLP course involved building a basic text processing module that implements sentence segmentation, tokenization, stemming /lemmatization, stopword removal, and some aspects of spell check. This module involves implementing an Information Retrieval system using the Vector Space Model. The same dataset as in Part 1 (Cranfield dataset) will be used for this purpose. The project is split into two components - the first is a *warm-up* component comprising of Parts 1 through 4 that would act as a precursor for the second and main component, where you improve over the basic IR system.

[Warm up] Part 1: Working out a toy IR system [Numerical]

Consider the following three documents:

d_1: Herbivores are typically plant eaters and not meat eaters
d_2: Carnivores are typically meat eaters and not plant eaters
d_3: Deers eat grass and leaves

1. Assuming {are, and, not} as stop words, arrive at an inverted index representation for the above documents.

Herbivores	Doc1
Carnivores	Doc2
typically	Doc1, Doc2
plant	Doc1, Doc2
meat	Doc1, Doc2
eaters	Doc1, Doc2
Deers	Doc2
eat	Doc2
grass	Doc2
leaves	Doc2

2. Construct the TF-IDF term-document matrix for the corpus $\{d_1, d_2, d_3\}$.

Terms \ Docs	**Doc1**	**Doc2**	**Doc3**
plant	0.97494	0.097494	0.0
eaters	0.194988	0.194988	0.0
grass	0.0	0.0	0.396241

deers	0.0	0.0	0.396241
meat	0.097494	0.097494	0.0
leaves	0.0	0.0	0.396241
carnivores	0.0	0.264160	0.0
typically	0.097494	0.097494	0.0
herbivores	0.264160	0.0	0.0
eat	0.0	0.0	0.396241

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

Retrieved Documents: Doc1, Doc2

4. Find the cosine similarity between the query and each of the retrieved documents. Is the result desirable? Why?

Cosine Similarity calculations:
Query vector = [0.54930614 0.54930614 0. 0. 0. 0.
0. 0. 0. 0.]
After the query vector consideration:
Doc1 vector = [0.09749375 0.1949875 0. 0. 0.09749375 0.
0. 0.09749375 0.26416042 0.]
Doc2 vector = [0.09749375 0.1949875 0. 0. 0.09749375 0.
0.26416042 0.09749375 0. 0.]
Doc3 vector = [0. 0. 0.39624063 0.39624063 0. 0.39624063
0. 0. 0. 0.39624063]
Cosine similarity values:
Doc1 : 0.56015692
Doc2: 0.56015692
Doc3 = 0

Ranking documents: Doc1, Doc2, Doc3

Is the ordering desirable? If no, why not?:
Yes the ordering is desirable and the ordering between the doc1 and doc2 can be altered as per wish as they have the query terms in them

[Warm up] Part 2: Building an IR system [Implementation]

1. Implement the retrieval component of the IR system in the template provided. Use the TF-IDF vector representation for representing documents.

The information retrieval has been implemented in the informationRetrieval.py file.

```
class InformationRetrieval():
    def rank(self, queries):

        tf_idf_q_sklearn = vectorizer.transform(query_words)
        # print("=============query_tf done=============")
        # tf_idf_q =  tf_q * np.log((len(doc_words)) / (idf + 1e-9))
        # print("=============query_tfidf done=============")

        tf_idf = tf_idf_sklearn.todense().T
        tf_idf_q = tf_idf_q_sklearn.todense().T

        norm_d = np.linalg.norm(tf_idf, axis=0).reshape(-1,1)
        norm_q = np.linalg.norm(tf_idf_q, axis = 0).reshape(-1,1)
        print("tf_idf_q.T: ",tf_idf_q.shape)
        print("tf_idf: ", tf_idf.shape)
        print("norm_q: ", norm_q.shape)
        print("norm_d.T: ",norm_d.T.shape)
        cos_sim = tf_idf_q.T@tf_idf/(norm_q@norm_d.T + 1e-7)
        print("=============cosine done=============")

        ranks = []
        for i in tqdm(range(len(cos_sim))):
            rank = np.argsort(-cos_sim[i,:], axis=1)
            rank = [self.doc_IDs[rank[0, j]] for j in range(rank.shape[1])]
            ranks.append(rank)

        doc_IDs_ordered = ranks
        return doc_IDs_ordered
# class InformationRetrieval():
```

[Warm up] Part 3: Evaluating your IR system [Implementation]

1. Implement the following evaluation measures in the template provided (i). Precision@k, (ii). Recall@k, (iii). $F_{0.5}$ score@k, (iv). AP@k, and (v) nDCG@k.

Precision@k: 0.635, 0.55, 0.48, 0.43, 0.39, 0.36, 0.34, 0.32, 0.301, 0.28

Recall@k: 0.109, 0.180, 0.229, 0.264, 0.294, 0.324, 0.349, 0.368, 0.388, 0.401

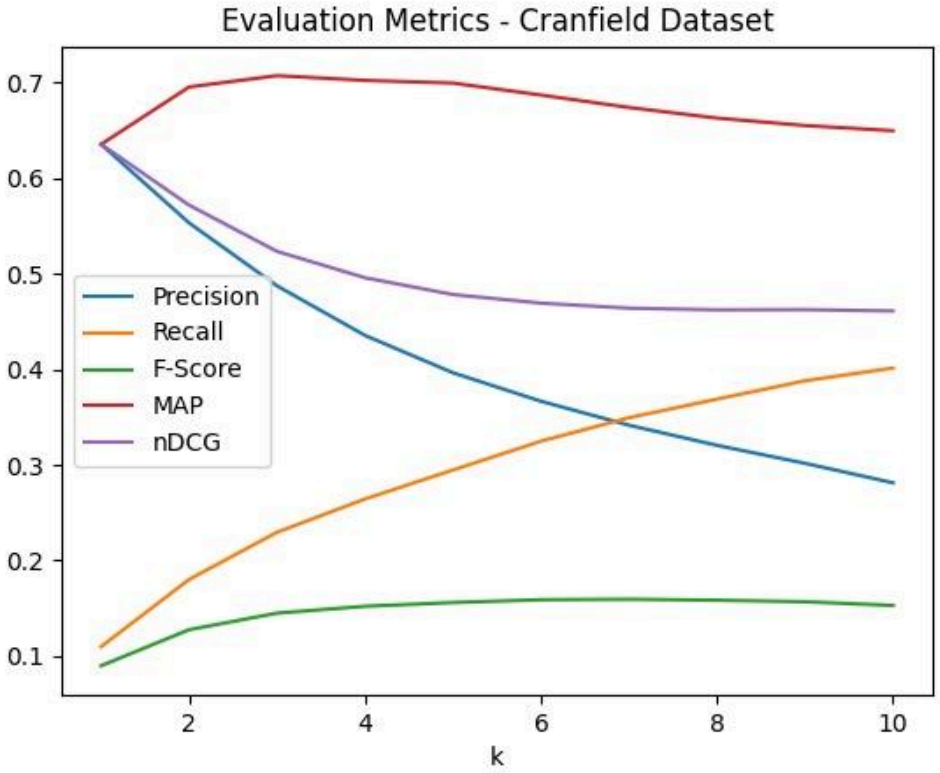
$F_{0.5}$ score@k: 0.089, 0.127, 0.144, 0.152, 0.156, 0.159, 0.158, 0.156, 0.153

AP@k: 0.635, 0.695, 0.707, 0.702, 0.699, 0.687, 0.674, 0.662, 0.655, 0.649

nDCG@k: 0.635, 0.571, 0.523, 0.495, 0.478, 0.469, 0.464, 0.462, 0.462, 0.461

2. Assume that for a given query, the set of relevant documents is as listed in incran_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. The 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.

Graph:



Observation:
From the plots, it is clear that the precision decreases with increasing the number of retrieved documents under consideration. We can also infer that the recall increases with k whereas the Fscore increases and then flattens out. MAP score increases for smaller values of k after which it starts to decrease. nDCG decreases in a plateauing manner.

3. Using the `time` module in Python, report the run time of your IR system.

24.8663 secs

[Warm up] Part 4: Analysis [Theory]

1. What are the limitations of such a Vector space model? Provide examples from the cranfield dataset that illustrate these shortcomings in your IR system.

Limitations:

1) Vector space model treats the terms as **bag of words** disregarding the context or meaning of the words or order of the words.
2) Due to higher dimensions, the individual term's dimensions diminish and they eventually are of less significance. Effectiveness of the word is reduced.
3) Resource intensive leading to scalability issues for larger corpus.
4) Semantic importance is significantly less.

Examples from your results:

1. For example in queries like 'Apple computers or mobile phones', 'boundary layer', etc, the documents retrieved would be based on terms like 'apple' and 'computers' or' boundary' and' layer'. The order or compound set of words will not be considered.
2. Heart attack and cardiac arrest are similar but VSM might not capture this relationship between them due to lack of semantic understanding.s

Part 4: Improving the IR system

Based on the factual record of actual retrieval failures you reported in the assignment, you can develop hypotheses that could address these retrieval failures. You may have to identify the implicit assumptions made by your approach that may have resulted in undesirable results. To realize the improvements, you can use any method(s), including hybrid methods that combine knowledge from linguistic, background, and introspective sources to represent documents. Some examples taught in class are Latent Semantic Analysis (LSA) and Explicit Semantic Analysis (ESA).

You can also explore ways in which a search engine could be improved in aspects such as its efficiency of retrieval, robustness to spelling errors, ability to auto-complete queries, etc.

You are also expected to test these hypotheses rigorously using appropriate hypothesis testing methods. As an outcome of your work, you should be able to make a statement of structure similar to what was presented in the class:

> An algorithm $\boldsymbol{A_1}$ is better than $\boldsymbol{A_2}$ with respect to the evaluation measure $\boldsymbol{E}$ in task $\boldsymbol{T}$ on a specific domain $\boldsymbol{D}$ under certain assumptions $\boldsymbol{A}$.

Note that, unlike the assignment, the scope of this component is open-ended and not restricted to the ideas mentioned here. For each method, the final report must include a critical analysis of results; methods can be combined to come up with improvisations. It is advised that such hybrid methods are well founded on principles and not just ad hoc combinations (an example of an ad hoc approach is a simple convex combination of three methods with parameters tuned to give desired improvements).

You could either build on the template code given earlier for the assignment or develop from scratch as demanded by your approach. Note that while you are free to use any datasets to experiment with, the Cranfield dataset will be used for evaluation. The project will be evaluated based on the rigor in

methodology and depth of understanding, in addition to the quality of the report and your performance in Viva.

Your project report (for Part 4) should be well structured and should include the following components.

1. An introduction to the problem setting,
2. The limitations of the basic VSM with appropriate examples from the dataset(s),
3. Your proposed approach(es) to address these issues,
4. A description of the dataset(s) used for experimentations,
5. The results obtained with a comparative study of your approach has improved the IR system, both qualitatively and quantitatively.

The latex template for the final report will be uploaded on Moodle. You are instructed to follow the template strictly.