# CS6370: Natural Language Processing
## Project

Release Date: 21st March 2024 Deadline:


| Name: | Roll No.: |
|---|---|
| Hari Charan | CS20B086 |
| Shashank Kakunuri | CS21B038 |
| Aditya Viraj | CS20B005 |


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.

To create an inverted index representation for the given documents, we need to tokenize each document, remove stop words, and then create a mapping of each term to the documents in which it appears.

inverted index representation:

| **Word** | **Documents** |
|---|---|
| herbivor | $d_1$ |
| typic | $d_1$, $d_2$ |
| plant | $d_1$, $d_2$ |
| eater | $d_1$, $d_2$ |
| meat | $d_1$, $d_2$ |
| carnivor | $d_2$ |
| deer | $d_1$ |
| eat | $d_3$ |
| grass | $d_3$ |
| leav | $d_3$ |

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

| Terms | d1 | d2 | d3 | doc freq | IDF | Query | weight1 | weight2 | weight3 |
|---|---|---|---|---|---|---|---|---|---|
| herbivor | 1 | 0 | 0 | 1 | 0.477 | 0.000 | 0.477 | 0.000 | 0.000 |
| typic | 1 | 1 | 0 | 2 | 0.176 | 0.000 | 0.176 | 0.176 | 0.000 |
| plant | 1 | 1 | 0 | 2 | 0.176 | 0.176 | 0.176 | 0.176 | 0.000 |
| eater | 1 | 1 | 0 | 2 | 0.176 | 0.176 | 0.176 | 0.176 | 0.000 |
| meat | 1 | 1 | 0 | 2 | 0.176 | 0.000 | 0.176 | 0.176 | 0.000 |
| carnivor | 0 | 1 | 0 | 1 | 0.477 | 0.000 | 0.000 | 0.477 | 0.000 |
| deer | 1 | 0 | 0 | 1 | 0.477 | 0.000 | 0.477 | 0.000 | 0.000 |
| eat | 0 | 0 | 1 | 1 | 0.477 | 0.000 | 0.000 | 0.000 | 0.477 |
| grass | 0 | 0 | 1 | 1 | 0.477 | 0.000 | 0.000 | 0.000 | 0.477 |
| leav | 0 | 0 | 1 | 1 | 0.477 | 0.000 | 0.000 | 0.000 | 0.477 |

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

Retrieved docs :- **1 and 2** will be retrieved because their vector space representation has dimen- sions of "plant" and "eater" that match the query's dimensions.

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

**Cosine Similarity calculations:** d1 = 0.327, d2 = 0.420, d3 = 0

**Ranking documents: 2, 1, 3**

**Is the ordering desirable? If no, why not?:** Not desirable, Document D3 appears to be relevant to the query because deer are plant eaters. However, using TF-IDF, this document is not retrieved.

[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.

| Code in InformationRetrieval.py |
| --- |

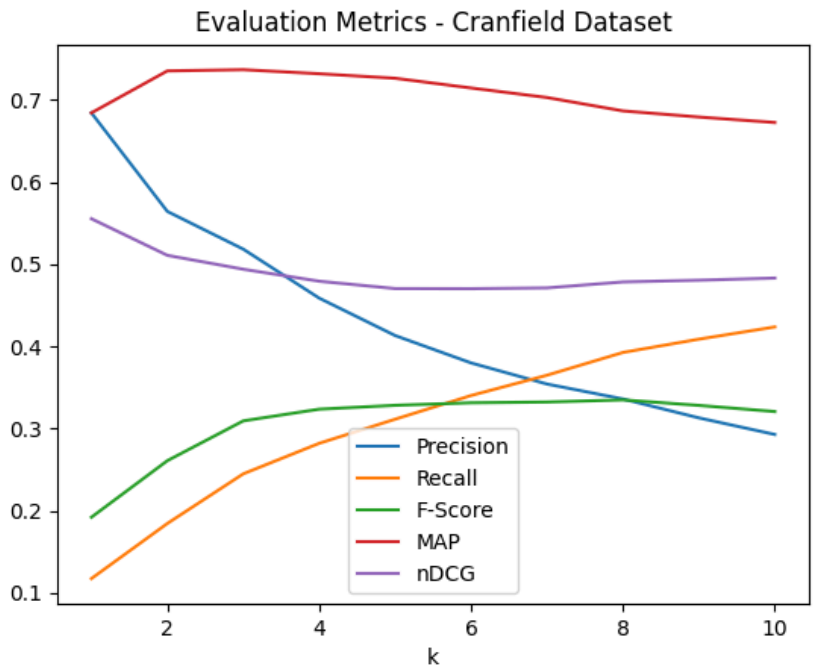
[Warm up] Part 3: Evaluating your IR system [Implementation]

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:**<br><br>**Recall@k:**<br><br>**$F_{0.5}$ score@k:**<br><br>**AP@k:**<br><br>**nDCG@k:**<br><br>All implementations in evaluation.py |

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:** The above graph is the moving averages of various evaluation measures for each consecutive rank k.

1. The IR system is effective when precision decreases as rank increases. As you move down the ranked list, the diversity of documents may increase. While some of these diverse documents may still be relevant, they might not match the user's query as closely as the top-ranked ones, resulting in lower precision.
2. Recall is found to rise monotonically as k increases, also expected when more documents are retrieved as more relevant documents are retrieved.

3. The harmonic mean of recall and precision is the F-score. The f-score initially rises with k, and the graph becomes flat with rising k, according to the precision and recall graphs.
4. It can be observed that Mean Average Precision increases at first, reaches a maximum, and then starts to decrease. This follows the precision curve.
5. Hence, all nDCG calculations are relative values of 0.0 to 1.0 and are similar across queries. At bigger values of k, there is a slight decrease in the nDCG values, which otherwise appear to remain relatively constant.

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

Using the time module it took **4.55 seconds** to execute from rank 1 to 10.

[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:**

**Bag of Words:** VSM treats documents as bags of words, disregarding grammar, word order, and semantics. This can lead to issues with understanding context and meaning.

**Lack of Semantic Understanding:** VSM doesn't have a built-in understanding of concepts or semantics, which means it may struggle with tasks requiring deeper comprehension beyond simple keyword matching.

**Sparsity:** In large document collections, the vector representations can become extremely sparse, making it computationally expensive and memory-intensive to store and process them.

**Examples from your results:**

**Bag of Words:** Consider a query about "wings" in the context of aircraft design. A document discussing "bird wings" might contain the term "wings" but is irrelevant to the query. However, in a bag-of-words representation, both documents containing "bird wings" and "aircraft wings" would be treated similarly based solely on the presence of the term "wings."

**Lack of Semantic Understanding:** Suppose a user's query is about "aircraft safety regulations." While there may be relevant documents in the Cranfield dataset discussing "aviation safety guidelines" or "aircraft regulatory compliance," VSM might struggle to retrieve these documents effectively due to lexical variations or differences in terminology, even though they are conceptually related to the user's query. VSM lacks the ability to understand the semantic relationships between terms beyond simple keyword matching.

**Sparsity:** Suppose the Cranfield dataset contains a large number of documents, each containing only a few unique terms. As a result, most document vectors will be highly sparse, making it challenging to accurately assess document similarity or relevance using traditional cosine similarity or other distance measures.

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.