Statistical Natural Language Processing

[Final Project Report]

Akshay Joshi
2581346
s8akjosh@stud.uni-saarland.de

Ankit Agrawal
2581532
s8anagra@stud.uni-saarland.de

Abstract:

The task is to build a multi-stage information retrieval system which accepts a list of queries and return the top K ranked documents for each query from the text corpus. Further, the system returns a list of top ranked passages/answer spans retrieved from the corpus of relevant documents.

a. Introduction:

1.1. Information Retrieval with Statistical NLP

From recent times, Natural Language Processing Systems have been enjoying wide popularity in various domains such as in the development of high precision Search Engines, Text Entity Recognition, Content Recommenders, Speech Recognition Systems and so on. The prominent usage is in the fields of Information Retrieval and Data Mining through the usage of sophisticated Language Models. In this project the scope is only limited to statistical and classical methods rather than deployment large-scale deep learning solutions. In this project, we deploy an array of both simple and advanced Information Retrieval systems which deals with efficient and robust organization, storage, retrieval & evaluation of information from text corpus/document evidences. IR systems are very widely used in everyday lives in the forms of search engines, files explorers, document searching tools etc. Thus, development of such high impact systems is desired. However, there are a multitude of challenges while exploring the vastness of natural language text and it's usage in the corpus scraped from the web.

The goal of a good IR system is to achieve high precision for a user query. But, at the same time, the system has to balance the trade-off to achieve respectable recall too. The system demonstrates high precision if the documents retrieved answers the user query completely i.e. all the relevant documents are ranked higher than the rest in the corpus. Whereas, recall is the fraction of the relevant documents that are successfully retrieved.

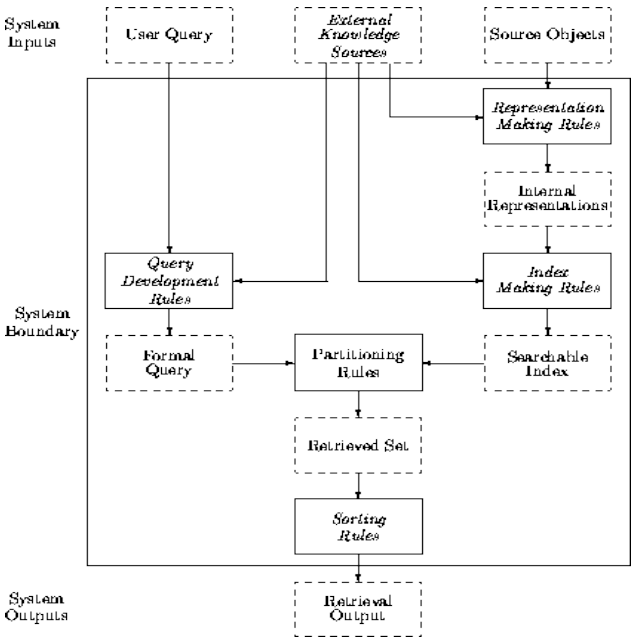

Fig: Block diagram of a typical information retrieval system

2. Task 1: Data Pre-Processing & Baseline Document Retrieval Model

Before we proceed to the development of a TF-IDF value based IR model. The task was to parse the provided text corpus scrapped from multiple sources such as LA Times, Financial Times and so on and create corresponding document evidences. The text in the corpus is passed through custom designed multiple filters to remove unnecessary punctuations, symbols and words. Later, the rest are tokenized and written to respective documents evidence ".txt" files. Similar strategy is applied to the test questions xml file as well to retrieve all the 100 test queries.

There are multiple challenges and tradeoffs while executing this pre-processing pipeline and if done incorrectly the overall model precision of the system suffers. More details are discussed in the next section.

TF-IDF based IR model consists of the following modules:

- Function to parse document evidences
- Function to parse queries
- Model to match query terms to the text in documents
- Document ranker using similarity functions

2.1.Model Architecture

The TF-IDF based IR system utilizes the following terminologies to retrieve desired documents for any query.

- Term Frequency: The total occurrences of a specific word in each document in a corpus is divided by the total number of words in the document.
- Inverse document Frequency: Log of the number of documents divided by the number of documents which contain the word w.
- TF-IDF: Product of both TF & IDF values.
- Cosine Similarity: The degree of similarity of a document vector to a query vector is normally cosine of the angle between them.

2.2. Results

The Mean Precision @ 50 is: 0.0847999999999999 (**Precision: ~8.5%**)

3. Task 2: Data Pre-Processing & Advanced Document Retrieval Model

The baseline VSM Model has many limitations thus we explore Okapi BM25 and BM25 Plus in this task. Okapi BM25 is a probabilistic retrieval model. The algorithm is used for ranking the documents. Given a query and set of documents, the model returns the ranked documents as per their distance to the actual query.

3.1. Results

The Mean Precision @ 50 for BM25 is: 0.107399999999999 (**Precision: ~11%**)

```
[INFO] Please be patient, building a massive postings list!
The Mean Precicion @ 50 is: 0.0847999999999999
The Mean Precicion @ 50 for BM25 is: 0.10739999999999995
MRR for BM25 is: 0.15448392574224493

Process finished with exit code 0
```

4. Task 3: Sentence Ranker & MRR

MRR is computed for analyzing a model that returns a list of relevant documents/answer spans for a set of queries, which then is ordered by the probability of their correctness.

4.1. Results

MRR for BM25 is: 0.1544839257422

5. Novel Contributions to the project

Tokenization: Used regex based tokenization to handle '-', """ and handle digits explicitly. The regex rules are mentioned below:

```
self.regex_str = [
            r"(?:(?:[a-z]|[0-9])+['\-_]+(?:[a-z]|[0-9])+['\-_]*(?:[a-z]|[0-
9])*)", # words with - and '
            r'(?:(?:\d+,?)+(?:\.?\d+)?)',  # numbers
            r'(?:[\w_]+)',  # other words
            r'(?:\S)'  # anything else
```

- It will handle the keywords like "mid-90's".
- Punctuations from nltk including some manual punctuations tokens were removed.

6. Conclusion & Future Work

Finally we have developed multiple models to solve the issues of IR and have optimized the systems to achieve robust Precision @ K and MRR values. As future work, we may use advanced neural methods with pre-trained word vectors and attention by transforming documents into feature vectors considering advanced features that takes account of semantics, long term dependencies and so on.