COL764-Project Milestone 3: Detailed submission with technical details of the approach, preliminary results

Team: Kota Factory

Manoj and Dipanshu

As mentioned in prior milestones, in this project we have two tasks:

Task 1: Identify relevant prior cases: The first task was to identify relevant prior cases for a given situation. There are 2914 case documents that were judged in the Supreme Court of India. For each query, the task was to retrieve the most similar/relevant case document with respect to the situation in the given query.

We implemented this task into following subtasks:

- **Data pre-processing:** In this step, we took care of the lower- and upper-case letters, stop word removal, stemming and removed punctuation and numbers from the case documents.
    - For this firstly we read text from each file, then remove punctuations with the help of self implemented function **(remove_punctuation),** then we tokenize this processed text with help of inbuilt split function. Then we use **Porterstemmer** for stemming and get the final processed text where all words are lower case,stemmed.
    - This all helper functions are in helper.py file, we use get_words function to process each file and get the processed text from the files.
- **Similarity score using TF-IDF:** we make tf_idf vectors of all files and the queries and then ccompute the cosine similarity between them.
    - for tf_idf vectorization, we have a dictionary called **allwords** , this dictionary contain all the word present in all files(i.e. vocabulary) , here key is a word and corresponding value is number of documents in dataset that contain that word . This will give us the idf of each word.
    - For tf, we make a dictionary corresponding to each file, each of this dictionary is (key:word,value:number of times this word appears in this document), so we get tf for each word in each file.
    - We just multiply tf and idf to get tf_idf vector for each file.
    - We do same for each query to get tf_idf vector for each query.
    - We then use helper function **getCosine** to get the cosine similarity between any two vectors(here we have dictionaries).

- For each query we get its cosine similarity with each document.
- **Formula to find cosine similarity and tf-idf:**

$$TF - IDF_i = TF_i * \log \frac{N}{DF_i}$$

$$similarity = cos(\theta) = \frac{A.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **BM25:** Again, we use self implmented helper function to get BM25 score of each query with all the documents.
  - Helper function is **getBM25,** avgerage doc length was computed when we were processing the documents, rest is the formula for BM25.
  - **Formula to calculate BM25 of a document:**

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

- For results, we find results for three typees of retrivals namely, TF-IDF,BM25 and combination of these two, i.e. producrt of BM25core and cosine similarity.
- To retrieve the documents with the highest rank in both TF-IDF and BM25, we multiply the cosine similarity scores of a query-document pair with its BM25 score.
- Then we sort the document according to thier BM25*(tf_-df score), for each query and rank them in order of relevance.
- **Word Embeddings:** We plan to use doc2vec/sent2vec embedding for the dataset as it is publicly available. We have not done this yet, but we will do this in our next submission.
- **SBERT:** We plan to use Sentence-BERT to understand the context of a query with other documents. We have not done this yet, but we will do this in our next submission.
- **Results:** we present the results for all three types of retrival techniques mentioned above.
- **In the results,** we print names of top 10 most relevant prior cases(i.e with highest similarity score) , also we print the ranks of the ground truth cases in our retrival.
- **The results are best for bm25 retrival.**
- **After this we will re-rank these retrieved relevant prior cases.**

Task 2: Identify the most relevant statutes for a given situation: The second task was to identify relevant statutes for a given situation. There are 197 statutes (Sections of Acts) from Indian law. For each query, the task was to retrieve the most similar/relevant statutes with respect to the situation in the given query.

We implemented this task same as the task one, the only differnce is instead of prior cases we use statutes for this task.

After this retrieval, we will use relevance models of Lavrenko and Croft's relevance models with a Unigram model with Dirichlet smoothing for reranking this retrieved relevant prior cases and statutes.