Evidence Retrieval for Fact Verification

INFORMATION RETRIEVAL

Project Final Term

Spring Semester - 2022

<u>Group 01</u>

Md Ashik Khan (21CS60A02)

Razorshi Prozzwal Talukdar (21CS60A03)

Major Kunal Pilley (21CS60D04)

Major Shreyyash Sharma (21CS60D06)

Indian Institute of Technology, Kharagpur

PROBLEM STATEMENT

1. **Fact Verification (FV)** is a challenging task which requires retrieving relevant evidence from plain text and using the evidence to verify or refute a given claim. Many claims require to simultaneously integrate and reason over several pieces of evidence for **Verification**. Fact or claim verification is a two-step process.

   a. Firstly, we need to retrieve supporting or refuting evidence related to a claim.

   b. Then based on the set of evidence snippets, the task is to determine whether the claim is true or false. In this project, we are interested in the first step, i.e., evidence retrieval.

RELATED WORK

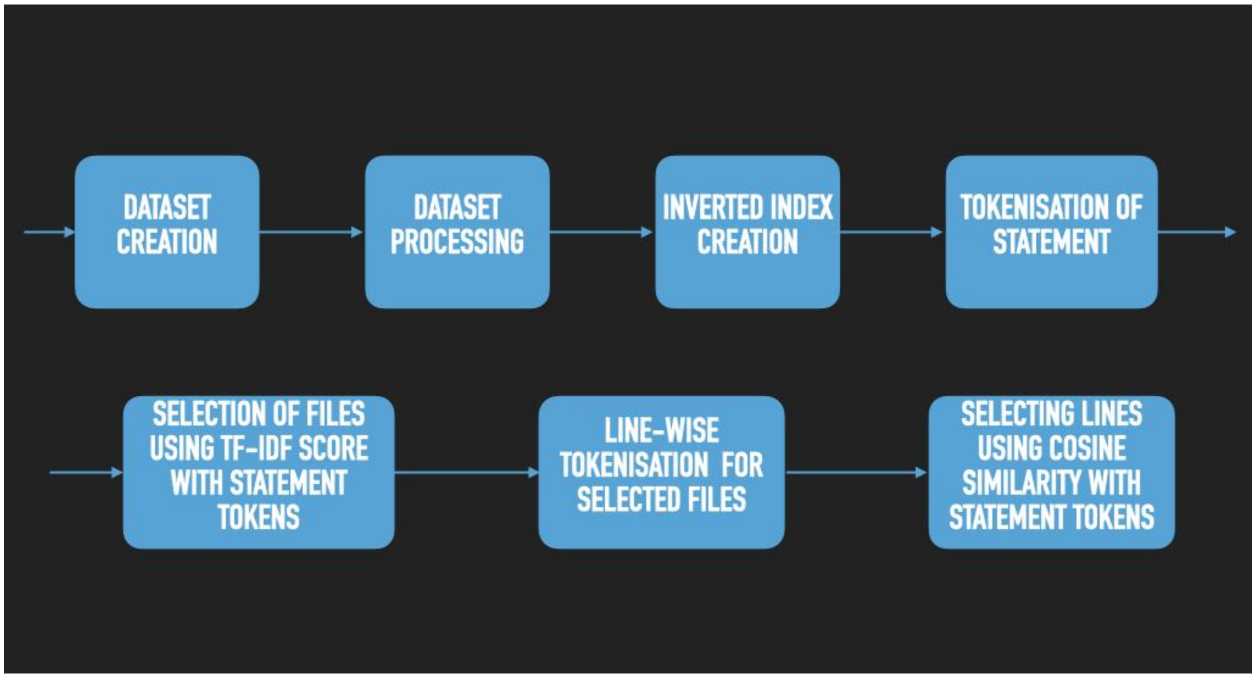
2. The reference data set that acts as a baseline for implementation is the FEVER dataset , which is a popular dataset based on 50,000 wiki pages and has around 1.8 lakh claims which are annotated as verified/ refuted/ not enough info.

3. The main philosophy behind the FEVER dataset is that they took random claims out of the Wikipedia pages and modified/ merged multiple sentences to form the claims against which they would work. The same was carried out with the help of annotators The annotators were asked later to label each individual claim generated as SUPPORTED, REFUTED or NOTENOUGHINFO.

NOVEL APPROACH

4. We propose a different approach to the issue and don’t use any questions which are based on the underlying dataset but attempt to verify/ refute any general query one might have. In this Manner the the Fact verification database is not limited to the 50,000 wikipedia documents as done in the FEVER dataset but subjected to the document handling capacity of the underlying architecture/system.

Flow of the Project


DATASET CREATION
DATASET PROCESSING
INVERTED INDEX CREATION
TOKENISATION OF STATEMENT
SELECTION OF FILES USING TF-IDF SCORE WITH STATEMENT TOKENS
LINE-WISE TOKENISATION FOR SELECTED FILES
SELECTING LINES USING COSINE SIMILARITY WITH STATEMENT TOKENS

Distribution of Work

5. The project was divided into the following tasks and the distribution of labour is as follows:

Serial	Task	Responsibility	Additional Support
1.	Database creation	Kunal Pilley	
2.	Documents pre-processing	Md Ashik Khan	
3.	Inverted index creation	Razorshi Talukdar	
4.	Fact retrieval	Shreyyash Sharma	Help in re-writing all the code in memory optimised manner

TECHNIQUES AND EXPERIMENT

Database Creation

6. We have used the WIkipedia xml dump of the most popular pages which can be found at https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2. Initial trials carried out on the full Wiki dump of English articles proved to be too expensive due to the limited resources available and the dump of popular articles was used instead.

7. Initially, we used PLY (Python Lex Yacc) to extract the articles from the xml dump, but due to the general nature of the parser and its slow speed, the parser was dropped. We then tried to use an xml parser lxml but did not get much success. We finally made headway using Beautiful Soup.

8. After extracting our articles, the large quantity of articles made it difficult to work and we had to use a smaller wiki dump of popular articles.

Documents Pre-processing

9. From the collected dataset , we created the article list in lexicographical order.

10. Collated Data. Convert the article text into string . Tokenize the string. Collect all the article names along with a unique delimiter.

11. Tokenize Article String

    a. The function accepts the text data as a parameter and does the following operations:
    b. Removes the punctuations from the text data
    c. Removed Stop Words: At first, reads the common English stop words from the file stopwords.txt and then returns them as a list. The function takes in tokenized words and removes stop words from it and the words which are smaller than 2 characters in length.
    d. Stem Words: The function takes in a list of tokens and returns a list of stemmed words using NLTKs PorterStemmer.
    e. Then, Store string and token along with the article file name and file no. We prepared the Collated Data for search and index preparation.

12. Challenges faced and Addressal

    a. It took roughly 6 hours of preprocessing time
    b. Produced Approx 70,000 file to collated data
    c. Cleanup Approx 70,000 file and tokenize

Index Preparation

13. Use of Tokenized Articles: For preparing inverted index we used the collected data file, and Collect all tokens from Colleted Data file

14. Create a set of tokens

    a. Remove duplicate tokens: There were so many duplicate tokens on the collected data file. At first, we removed all duplicate tokens.
    b. Count tokens by document terms
    c. Store with lexicographical order: Store all tokens in alphabetical order.

15. Store the TF at the time of Creating Inverted Index: The function takes in a list of tokenized words and returns a dictionary with token as the key and the tf_score for that token as the value.

16. File size of Inverted Index: 64.7 MB

17. Static Database

    a. We used Static Database.

    b. Using that database reduces the search time up to 30 sec.

Fact Retrieval

18. After creation of the articles pre-processed database and the inverted index, we entered the last stage of the project, retrieving the relevant sentences.

19. We first loaded the pre-processed articles and the inverted index, since we worked on a static dataset, we didn't need to worry about updating our inverted index.

20. The query was retrieved from a file query.txt and passed through the nltk library where the following tasks were performed.

    a. Tokenize and remove punctuations
    b. Remove stop words from the given query
    c. Stem the word tokens
    d. Store the query stemmed tokens along with their frequency

21. Use the inverted index and the query tokens to prepare a list of documents to be searched

for the relevant text. The index created was then arranged in decreasing order and the top 25 documents were chosen for retrieval. The underlying principle of ranking a document was as follows:

*Rank += (tf * frequency of token in query)/(length of posting list in inverted index)*

22. After identifying the documents, all the sentences of the selected documents were tokenized and matched against the query tokens using a modified *Cosine Similarity* formula. The formula used to keep a score for each sentence is as follows:

Score += (|freq of sentence token|.|freq of query token|)/(len of the sentence tokens)

23. The length of the query was not taken into consideration since it is a constant multiplicative factor for each sentence and doesn't affect the ranking of the sentences. The primary aim of the modification was to keep computations required to a bare minimum.

24. The ranked sentences are then displayed to the user in a descending order of score.

ANALYSIS AND FUTURE THOUGHTS

25. Response Time

    a. The present response time is between 3-4 mins for each query, which is not acceptable on a modern system.
    b. The same can be potentially reduced to under 1 sec by changing the data storage format and applying more fine level control over the data pre-processing used and optimising the code further. The same is possible but requires a time investment of about a week with good computation resources.

26. Query Accuracy

    a. The dataset used at present is restricted to approx 70,000 wikipedia articles. Expanding this further will greatly improve the recall of the system.

27. Spell Correction

    a. A number of times, the user enters the wrong spellings, correction of the same may be considered to be added.

28. Compound Sentences

    a. The scope of the project was limited to a single sentence, the same may be expanded to include tokens not in the primary sentence from alternate sentences.

References:

1. Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In Proceedings of EMNLP, pages 534– 545.
2. Soleimani, Amir, Christof Monz, and Marcel Worring. "Bert for evidence retrieval and claim verification." *European Conference on Information Retrieval*. Springer, Cham, 2020.
3. Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of EMNLP, pages 632–642.
4. Soleimani, A., Monz, C. and Worring, M., 2020, April. Bert for evidence retrieval and claim verification. In *European Conference on Information Retrieval* (pp. 359-366). Springer, Cham.
5. Soleimani, Amir, Christof Monz, and Marcel Worring. "Bert for evidence retrieval and claim verification." In *European Conference on Information Retrieval*, pp. 359-366. Springer, Cham, 2020.
6. Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer opendomain questions. In Proceedings of ACL, pages 1870–1879.
7. Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017c. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In Proceedings of ACL, Workshop on Evaluating Vector Space Representations for NLP, pages 36–40.
8. Alexis Conneau, Douwe Kiela, Holger Schwenk, Lo¨ıc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In Proceedings of EMNLP, pages 670–680.
9. Zhang, X., Chen, M.H. and Qin, Y., 2018, September. NLP-QA Framework Based on LSTM-RNN. In *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)* (pp. 307-311). IEEE.
10. Nie, Y., Chen, H. and Bansal, M., 2019, July. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 6859-6866).
11. Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pages 534–545.
12. Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.