COMP90042 Project 2019: Automatic Fact Verification

Prajakta Jangle
936261

Tarun Dev Thalakunte Rajappa
934175

Abstract

The advantages of automatic fact verification are undeniable as they provide a faster and reliable alternative to validate information over human resources.(Hassan et al.2015). In this paper, we attempt to do so over the Fact Extraction and VERification (FEVER) dataset by presenting a comparison of two approaches for document and sentence retrieval - TD-IDF and Elastic Search. We compare how they perform during label classification using Random Forest and Multinomial Naive Bayes classifiers.

1 Introduction

The automatic fact verification task provides a set of claims to be verified against a series of documents provided in the FEVER dataset (Thorne et al.2018). The system is required to return the label of each claim amongst Supports, Refutes and Not Enough Info along with the sentence ids as evidence in case of the former two labels.

The project, thus, consists of two important components, -

- Information retrieval: Retrieving the relevant documents and sentences from the wiki set.
- Recognizing textual entailment between the claim and wiki texts

In this paper, we focus on each of these components by comparing various techniques to achieve better results. We begin by using the TF-IDF model for retrieval of evidence sentences and feature extraction, and then we use the readily available tool Elastic search to retrieve evidence and extract features. For each feature set obtained from the former two approaches, we use Random Forest and Multinomial Naive Bayes classifiers to predict the label for each claim. In conclusion, we use score.py to determine the label accuracy, Sentence F1 and document F1 scores for each set of results obtained.

2 Background

In this section, we introduce the concepts used in the development of the system.

2.1 Elastic Search

We use Elastic Search for information retrieval to improve over the TF-IDF model for a claim, thus leading to an increase in our Sentence F1 and label accuracy. Elastic search is an open source software that is used for text search and analytics built on Apache Lucene and uses the boolean model(Gormley and Tong2015).

2.2 Multinomial Naive Bayes

The initial approach towards building this system include using classifiers that are easier to train and quicker to compute. Hence, Naive Bayes was the building block towards the final system. As mentioned in (Gupte et al.), "If processing power and memory is an issue then the Nave Bayes classifier should be selected due to its low memory processing power requirements."

2.3 Random Forest

Random forest classifier is a supervised machine learning algorithm that is built on the concept of decision trees. The bagging method is used to build decision trees and all the predictions are aggregated together to achieve an accurate prediction. Random forest is widely used in text classification for its high performance in modelling time and predicting accurate results. Furthermore, it is a good candidate for the system as it can cope well with high dimension data(Svetnik et al.2003).

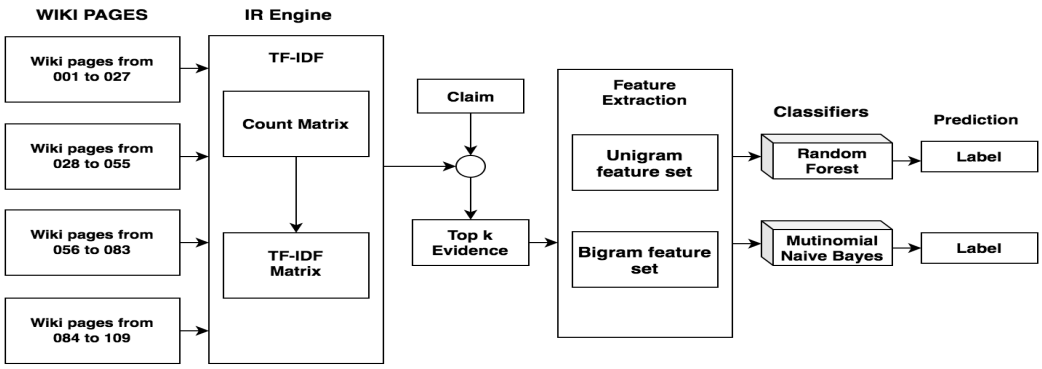

Figure 1: TF-IDF

3 System Design

The system can be broken down into two subsections. These subsections refer to the feature extraction method used for the system. These are namely, TF-IDF model and Elastic Search. Each subsection is used for label prediction and extracting the relevant evidence.

3.1 TF-IDF Model

The Term frequency - Inverse Document frequency matrix is built for the given Wiki pages file which consists of 109 documents. The steps for this are -

1. We preprocess the text further by removing the stop words and stemming using Porter Stemmer.

2. As mentioned in (Thorne et al.2018), we represent each word as a hash function by using the sklearn package to make it memory efficient to store words and easier to multiple matrices later. The function used is "murmurhash3_32" which stores data as 32 bit unsigned integers. All documents contained in the wiki pages are processed similarly to generate the term count matrix.

3. It is computationally expensive to process all the wiki pages together as it takes more time. Hence, we divide it into 4 batches with the first 3 batches containing 27 pages and the 4th batch containing 28 pages to create a batch term count matrix for each. The separate batches are then merged into a single count matrix.

4. The count matrix is then used to extract the TF-IDF matrix and stored as csr_matrix, thus making the storage memory efficient.

5. We create and store an index file for faster retrieval of sentences from the wiki pages text file. The wiki pages index file is a dictionary of document IDs for each document in wiki pages. Corresponding to each document ID, the index file stores the sentence number and line number in the wiki pages text file.

3.1.1 Document Retrieval

For retrieval of the relevant documents for a claim, it is imperative for them to be represented in the same format for comparison. Hence, each claim is preprocessed in a similar fashion by tokenizing, removing stop words and stemming text and used to generate the count matrix. Now that we have both the claim and documents as matrices, the relevant documents are found by performing dot product between them. The documents retrieved are sorted in descending of their similarity score and top 10 documents are considered.

3.1.2 Sentence Retrieval

Sentences in the top 10 documents are individually processed in the same manner as the documents. i.e., they are represented as matrices. The matrix multiplication among the claim and sentence yields scores for each sentence. The top 3 sentences are finally used as evidence for label prediction.

3.2 Elastic Search

Elastic Search is built as a local server for indexing the sentences in wiki pages. The index for each sentence in wiki pages is stored as a dictionary that contains the document ID, processed document ID, document sentence, processed document sentence and the sentence number. The processed

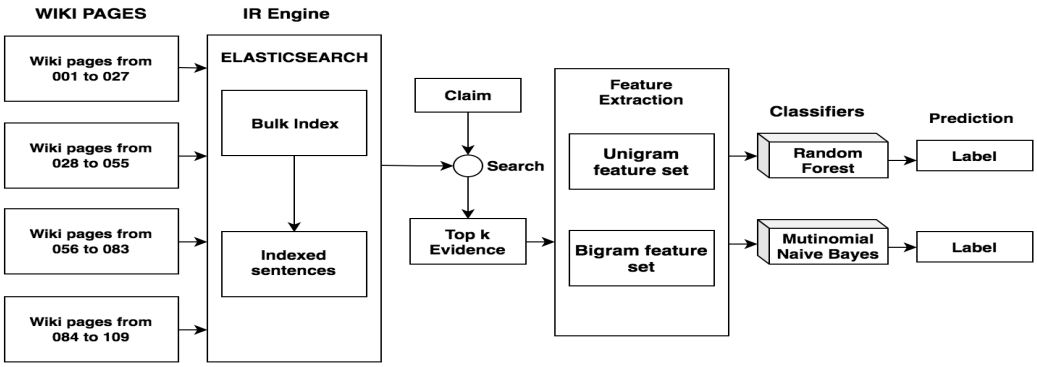

Figure 2: Elastic search

document is obtained by splitting the document ID on "_" and processed sentence is obtained by pre-processing through removal of stop words and stemming.

3.2.1 Sentence Retrieval

Elastic search indexes the wiki pages for each line. In order to make the retrieval more efficient the following steps were undertaken -

- A noticeable difference between the claim and wiki pages text included that the claim included the characters "(", ")" and ":" whereas wiki pages replaces the same with "-LRB-", "-RRB-" and "-COLON-" respectively. Hence, the claim was amended to match the format in wiki pages(Yoneda et al.2018). Boosting is a feature in elastic search where in priority can be given for particular fields by increasing their default values.

- Some claims included the document ID within the claim. Thus, the mentioned document IDs must be given higher importance. For instance, the relevant document ID The_Disaster_Artist-LRB-film-RRB- for the claim 2015 was the year when the principal photography of The Disaster Aritst (film) started. is ranked 4th in the list prior to boosting. The relevant document ID moved to the 1st position after implementing boosting. This was achieved by adding a boost factor of 20% for the document IDs and 5% is added for a match between the processed claim and processed sentence.

After the application of the above techniques, elastic search is used to retrieve the top 3 matching sentences.

3.3 Sentence Retrieval NOT ENOUGH INFO

Although the label should have no evidence attached to it, we need to add features for it in order for our model to train to classify the label correctly. Hence, we add the top 5 sentences as evidence for this label as well.

3.4 Label Prediction

After sentence extraction using TF-IDF and Elastic Search, these sentences are used to predict labels for a given claim. Random Forest and Multinomial Naive Bayes are two techniques used for doing so. The features for these classifiers are identified using the n gram language model.

3.4.1 Unigram features

Unigram features are calculated by matching claim tokens to the sentence tokens. Hence, the features are stored in the form of a dictionary with value = 1 in case of a match. This process is repeated for each claim.

3.4.2 Bigram features

The Bigram features are calculated by creating bigram pairs between the claim and evidence sentences by picking a token from each. We choose to store bigram features this way in order to incorporate data from both the files for better matching. The bigrams value is matched to 1 and added to the feature set.

After the calculation of feature sets for both n gram models, DictVectorizer is used to convert the dictionary values of each claim into Vector Space Model. These Vectors are fed into Random Forest and Multinomial Naive Bayes Classifiers to build the model and predict the label for both development set and the test set.

Table 1: Development set

		TFIDF			Elastic		
	Method	Accuracy	Doc F1	Sen F1	Accuracy	Doc F1	Sen F1
Unigram	RF	36.29	22.74	16.8	35.11	44.22	29.17
	MNB	36.83	22.74	16.8	35.77	44.22	29.17
Bigram	RF	41.63	22.74	16.8	38.77	44.22	29.17
	MNB	43.79	22.74	16.8	44.87	44.22	29.17

Table 2: Test set

		TFIDF			Elastic		
	Method	Accuracy	Doc F1	Sen F1	Accuracy	Doc F1	Sen F1
Unigram	RF	35.95	22.48	12.91	34.95	42.57	27.4
	MNB	37.05	22.48	12.91	35.43	42.57	27.4
Bigram	RF	39.3	16.89	11.86	41.30	43.95	28.81
	MNB	41.06	16.89	11.86	44.92	43.98	28.82

4 Results

Table 1 and table 2 show results for development set and test set respectively. Doc F1 and sen F1 refers to the document F1 score and sentence F1 score respectively. Some key obervations here are -

- TF-IDF approach to retrieve evidences performs poorly compared to Elastic Search.
- Document F1 score is significantly higher while using the Elastic Search approach for both development and test sets.
- Multinomial Naive Bayes(MNB) performs slightly better than Random Forest(RF) for both methods TF-IDF and Elastic search.
- The Random Forest takes significantly more time to build the model for bigram feature set as the dimension is very high compared to unigram feature set.
- The best performance is achieved by Multinomial Naive Bayes with a bigram feature set with a higher performance than the baseline of 40% label accuracy and 20% sentence F1 score.

5 Error Analysis

- Table 3 clearly shows that training set has a higher number of supported claims whereas the distribution is equal for the development set. In order to avoid bias of the system towards "SUPPORTS" label, 29775 instances were randomly selected for each label. This resulted in a slight increase in label accuracy.

Table 3: Label breakdown

	Train set	Dev set
SUPPORTS	80035	1667
REFUTES	29775	1667
NOT ENOUGH INFO	35639	1667

- Our present system returns just the top 3 relevant sentences depending on their Elastic Search score. However, for a number of claims, it is evident that the score significantly drops after the first or second retrieved sentence. For e.g., for claim "Murda Beatz is a hip hop record producer." the top 3 sentences were ("Murda_Beatz", 0), ("Ski_-LRB-record_producer-RRB-", 0) and ("Swizz_Beatz", 1) and respective scores are 116.57, 81.98 and 78.93. Thus, implementing a threshold dependent on the score of the top sentence could potentially improve the system by impacting the sentence F1 score.
- Our approach does not consider a chain of evidence in order to support or refute a claim. For instance, for the claim "Stan Beeman is only in shows on BBC." our system retrieves the following top 3 evidences ("Stan_Beeman", 0), ("Beeman_Hollow", 0), and ("Richard_Beeman", 2) which just matches the tokens, where as the actual evidences are interlinked.

6 Conclusion

We present a system in this paper with maximum performance higher than the baseline system with 44.92% label accuracy and 28.82% sentence F1 score achieved using a combination of Elastic Search and Multinomial Naive Bayes. Our system consists of 2 parts - Information retrieval using TF-IDF or Elastic Search, and Textual entailment using Random Forest and Multinomial Naive Bayes classifiers. Our results indicate that elastic search is a user-friendly tool that succeeds in increasing document retrieval in comparison to TF-IDF whereas multinomial Naive Bayes slightly outperforms Random forest for this data set. The system can be extended further using tools such as named entity tagging and deep learning techniques for label prediction.

References

Gormley, C. and Tong, Z. (2015). *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition.

Gupte, A., Joshi, S., Gadgul, P., and Kadam, A. Comparative study of classification algorithms used in sentiment analysis.

Hassan, N., Adair, B., Hamilton, J., Li, C., Tremayne, M., Yang, J., and Yu, C. (2015). The quest to automate fact-checking. *Proceedings of the 2015 Computation + Journalism Symposium*.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., and Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958.

Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). FEVER: a large-scale dataset for fact extraction and verification. *CoRR*, abs/1803.05355.

Yoneda, T., Mitchell, J., Welbl, J., Stenetorp, P., and Riedel, S. (2018). Ucl machine reading group: Four factor framework for fact finding (hexaf). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102.