Automatic Fact Verification System

Saransh Srivastava
Student ID: 1031073
saranshs@student.unimelb.edu.au
University of Melbourne

Waqar Ul Islam
Student ID: 1065823
islamw@student.unimelb.edu.au
University of Melbourne

I. Abstract

Automatic fact verification systems have gained a lot of interest in the academia in recent years. This report is inspired from the FEVER challenge implementation, presently going on as a CodaLab competition. In this report, we are going to explain how our system enables evidence extraction and verification of a claim using a wikipedia corpus as the knowledge source. We follow document retrieval IR step by keeping word counts in memory and top sentence extraction using similarity scores. We use a simple logistic regression as a baseline model and an artificial neural network as our predictor system. Our end-to-end system gives an label accuracy of 45.7

II. Introduction

The task of this programming assignment require finding evidences in the text corpus which relate to the claim query. Further based on these evidence set, predict if the evidence SUPPORTS or REFUTES or are in sufficient (NOT ENOUGH INFO) for the claim. As the data is nowadays is getting bigger and complex at the same time, manual techniques are no more feasible to verify data. Automation for this purpose is essential and we will try to implement a model which will help us understand the real-world problem and its fragments. The fact-checking has become increasingly important as it allows to verify controversial claims stated on the web. Research at a whole new level is being conducted in this field to find more improved and efficient methods. For this purpose, we have been provided with JSON format to train our model which contains 145449 claims and for the development purposes we used a subset of this train set which contains 5001 claims. Finally, after completing our model we will run the test-unlabeled file which contains 14997 claims to get the concluding results. We will extract evidence sets and predict labels. Every claim has a set of evidence, which is a collection of sentences from Wikipedia documents. On the basis of these sentences we have to label each claim as Supports; Refutes or Not Enough Info. To achieve the mentioned functionality, we created a model which can be divided into different parts. The first step is document retrieval, in this step we retrieved the documents which contains sentences related to the given claim. For the next step, we need to extract sentences from the retrieved documents. Then each of the evidence collected is passed on to a model which then predict each label for every claim according to the given evidences. To compare system performance, we compare the label accuracy, sentence F1 and document F1 scores using the true values. In this report, we describe our model that we developed to address the given task. The efficiency of our model depends upon the different methods which have been used to create this model, which we will present in detail in this report. In order to make this project more exciting, a project has been created on Code Lab by the name of Automatic Fact Verification, where we must submit our final code in order to compare the efficiency of our model with other teams.

III. Automatic Fact Verification System

In this section we will explain the design of our system and all the steps which were taken to implement the functionality.

A. Document Retrieval

This is the first step towards extraction of an evidence set for a given claim. As explained in the [System design] figure 1, we created a simple inverted index metric to retrieve top documents matching the claim. Each word in the matrix is associated with the count of its occurrence in the available documents. We also augmented our matrix with page identifier information to assign more weights to documents having majority of the related topic. We empirically deduced that, this is much faster than a tf-idf document retrieval method. In order to extract only relevant documents matching our claim, we clean the query by removing all stop words. Without making our document retrieval step computationally expensive, we extracted the documents by matching the filtered query as a whole and we also retrieved documents by matching the words in our query individually. With these two combinations we retrieved all the documents related to the query. This later helped us in extracting relevant sentences.

B. Sentence Selection

Our sentence selection was divided into two major parts. Since the document retrieval was in-memory, it was fairly fast. We realized from our training and development set, that claims are based around relationship between entities which can be uniquely identified.

1) Entity Identification: In the candidate documents, we gathered the page identification information and compared them with our clean claim query. Since most entities are uniquely identified with more than one word, we ranked the

search order of proper noun entities before nouns, keeping the order in which they appeared in the claim. Additionally, we combined proper nouns pair-wise and ran our search on page identifiers using it. *For Example*: *Sexual slavery is a reason for human trafficking.* It will look for page identifiers having *sexual_slavery* and *human_trafficking* before looking for *sexual or slavery or human or trafficking.* Another heuristic which helped us remove noise from our system was the removal of search child words given that the parent search word already extracted relevant sentences. So, our system avoids search sentences having *sexual and slavery* if we already recognized sentences having *sexual_slavery,* for example.

2) Sentence Scoring and Ordering: With our heuristic, we were able to filter our possible evidence sentences by just analyzing the meta data associated with these sentences. Once all these sentences are retrieved, we now rank them based on Wordnet's similarity scoring. In order to normalize the similarity between two sentences, we average the similarity values before scoring a claim-evidence pair. This approach helped us to boost computation speed as we only compare sentence similarity scoring on relevant sentences.

To order our top scoring sentences, we realized that sentences with higher normalized word count are more relevant to our claim. As a final filter and order, we used Jacobean similarity scoring. The passed sentences of our extraction system becomes the evidence set for the given claim.

Jacobean similarity scoring improved the filtration of our final evidences. Cases where normalized word count was high were now moved upwards in the rank with a better accuracy. Sentences such as *The only film ever created in 1997 was Soul Food.* were negated easily because system found the supporting evidence. It also correctly identified statements such as *Shawn Carlson is only German.* Another case where our system correctly identified the claim, where devset was wrong *PacSun sells products designed for humans aged 13 to 20 years old.* For this claim devset identified wrong evidences but our system was able to retrieve the correct ones.

C. Training

In our training model, we evaluated two models - as a baseline we used a simple logistic regression model to compare our results with our system of sequential neural network. Our logistic regression model used lbfgs solver and with an iteration of 1000. For our ANN model, we introduced a dense layer of 512 dimension size. We used a non-linear activation function to set negative values to zero and everything else to identity function. We made a 50 % previous output dropout. Our final prediction layer has an activation which sums together each of these outputs and normalize it to turn into a probability of the likeliness of the label category. We trained our system on each claim-evidence pair sentence by first combining them and then converting them into vocabulary vector space. We also encoded our labels before training out classifier for prediction.

Testing At testing time, we combined the output of each claim-evidence pair label and calculated the majority label assigned by our model. Based on the majority prediction for our claim, we predict the evidences related to that label as our evidence set.

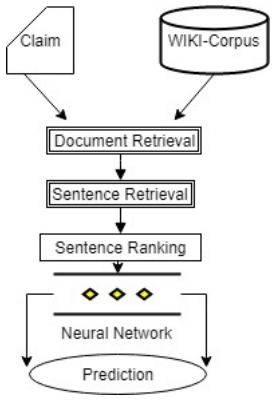

Fig. 1. System Design

IV. Result

The evidence generation took 23 hours for the test-unlabelled.json file when run on a MacBook Pro on a 2.3 GHz Intel Core i5 with 8 GB 2133 MH memory. The independence of evidence collection per claim gave us the opportunity to parallelize the process. The complete result analysis of our system can be obtained from TABLE 1. Our system predicts the FEVER like data set labels with 45.7 % accuracy.

TABLE I

Performance Parameter	Label Accuracy	Document F1 Accuracy
Codelab Test Output	45.7%	27.2 %
Our System - Devotes	43.81%	26.85%
Baseline - Devset	40.99%	25.98%

V. Error Analysis

In this section, we will present the error analysis for each of the subtasks of our fact verification system, the challenges and suggest areas of improvement

A. Document Retrieval

We realized that document retrieval step should be fast and flexible but without loss of relevant information. However, there were some areas we realised our system fail to capture. Since the system is not normalized on words, it becomes biased towards dominating words in the claim. For example - *Olivia Munn failed to be credited as Lisa Munn.* extracts documents related to *failed* and *credited* rather than relating to the two proper nouns. We partially overcame this by including both sentence level and word level most common document retrieval. In future, we wish to explore tf-idf based indexing which should be able to handle this problem better.

Our document retrieval also fails in cases where noun phrases are made of common nouns used as proper nouns because of the simplicity of the document retrieval method. Sentences like - "Julianne Moore was not in the television series As the World Turns." and "English people are disconnected from the Jutes and Frisians" does not retrieve correct documents, for example.

1) Sentence Retrieval: During relevant sentence retrieval, one of the primary problems our system face is that we collect first N sentences rather than top N sentences. This leads to problem when an entity is present many times in the relevant documents.

As a future implementation, we wish to explore combining both sub-tasks i.e. collect all relevant sentences and sort them according to scores assigned and analyse our system performance.

Another area of error is the normalization of claim queries. Consider a claim like - *Matthew Gray Gubler is a citizen of the United States of America.* Our system rates high score to sentences having *United States of America* rather than *Matthew Gray Gubler* giving irrelevant evidences for the claim.

Some other areas of error arise due to non-preprocessing of terms like *-LRB-*

and removing stop words in proper nouns such as in a title of movie. Further, we realised that instead of considering single words for search, we can also just have stopped at matching more than one word in the claim with page identifier.

2) Modeling: Our label prediction model is based on neural network but we realized that it does not always predict correct label even if claim and evidence are correctly identified as in the case of *John Krasinski is an astronaut*.

Additionally, our model predicts a label for each claim-evidence pair and then finds the majority label for a claim which might not be always correct. In cases where evidence set has correct evidence in minority will be dominated by incorrect evidences. We suspect that an ensemble system of low level and high level classifiers will be able to capture features better and predict more accurately.

VI. Future Work

The nature of problem is a general text classification problem based on question-answering methods. we realise that there is a continuous scope of improvement but given the time constraints of this semester project, we kept the scope of our implementation limited. As a future work, we would like to explore more heuristics on the dataset and use Stanford's natural language interface to capture features. We would also like to explore ways to expand page identification matching with content of sentences by augmenting page identification with key sentence information. Also, another interesting information to look at would be to search for noun phrases in the corpus extracted from retrieved sentences.

Our research suggests promising results in many papers using word embedding like GloVe and more complex word2Vec and Bert. Another promising direction to explore is using machine learning models for evidence feature extraction.

For training of model, we plan to use bidirectional LSTM layers which can take input not only from source but also from neighbouring LSTM layers for learning.

VII. Conclusion

In this report, we presented our fact verification system, which we developed as the final project of this course. According to the results our system was able to validate and negate the given claims through the WIKI corpus with an accuracy of 45.7 %. For the final assessment it took us 23 hours to verify all claims given to us in a JSON file. We tried to improve the performance of our system by parallelism the code. The system takes a WIKI corpus against which it verifies the submitted claims. An inverted index is being created of these documents. All the claims are entered into the system as an input. So, for the first step, relevant documents are retrieved by matching the claim query. The claim query is being filtered for any stop word. The query matches the whole words as well as parts of the query, once the documents are retrieved, relevant sentences are extracted from the documents. These sentences are then ranked on the bases of similarity score of Jacob algorithm and word net similarity score scheme. These sentences are then fed to the neural network, who predicts the final evidences and labels the claim. Our system handles a subset of the claims properly, however for the rest, our system needs further improvement, which we intend to in future.

VIII. Acknowledgment

We would like to express our appreciation to all those who provided us the possibility to complete this report. A special thanks to Professor Trevor Cohn, whose contribution in stimulating suggestions and guidance, helped us though out the project and special thanks to Shivashankar Subramanian for his constant support.

References

[1] Fever Challenge. https://www.aclweb.org/anthology/W18-5501