CSE 635 Natural Language Processing and Text Mining:

Fact Extraction and Automated Claim Verification

Team tmp
Anas Hamed
Anirudh Sridhar
Snigdha Gupta

Abstract

We develop a system for the Fact Extraction and VERification (FEVER) challenge. The challenge provides a large-scale dataset for the consecutive steps involved in claim verification, in particular, document retrieval, fact extraction, and claim classification. We implemented NER for document retrieval, Siamese network with a BiLSTM model for sentence selection, and an enhanced LSTM (ESIM) for recognizing textual entailment. The system proposed in this paper scored a FEVER score of 34.29 which places us in 44th position on the fever leaderboard.

1 Introduction

The increase in amounts of textual information available combined with the ease of sharing it through the web has increased the demand for verification, also known as fact-checking. An abundance of incorrect information can plant wrong beliefs in individual citizens and lead to a misinformed public, instigating chaos amongst the public, organizations, and countries (Thorne et al, 2018). In this context, technology to automate fact-checking and source verification is of great interest to both media consumers and publishers. In this paper, we propose a method to verify a claim with Wikipedia articles and find evidences to SUPPORT or REFUTE a claim.

Verification of any given claim starts with the retrieval of a number of documents from Wikipedia that are relevant to the claim. Once retrieved, a number of sentences is chosen from the documents as potential evidence candidates. Selection of sentences also takes into account the context of the claim. These sentences are then fed to the final stage of the system where a decision is made on the selected evidence about whether the claim can be logically entailed from this evidence. The FEVER task as a whole is challenging, it requires the system to not only perform well on textual entailment, but to also retrieve the necessary evidences from which the entailment should be made, which means that the performance of the entire system is penalized if irrelevant evidence is chosen.

In this work, we introduce three separate modules that tackle each of the three stages of the task separately. For document retrieval, a semantic approach was chosen wherein entities are extracted using Named Entity Recognition and Capitalized Expressions. These entities are then used to poll the Wikipedia database for matching articles, and the sentences are forwarded to the next stage. Sentence selection is carried out through encoding features relating sentences to the article, and feeding these features to a neural network, which in turn assigns a score for each of the sentences in terms of their suitability as evidence sentences. The top scoring sentences are selected, and run through the entailment system, which is implemented as an Enhanced Semantic Inference Model (ESIM) (Chen Q et al. 2017) coupled with a neural aggregation network. Each of the provided sentences are evaluated by the ESIM network, and the predictions are aggregated to draw a final conclusion on whether the evidence serves to support or refute the claim, or declare the evidence to be inconclusive in regard to the claim.

2 Related Work

The FEVER system encompasses three different components: Document Retrieval, Sentence (evidence) Selection, and Recognizing Textual Entailment. The FEVER Shared Task required participants to develop systems to label claims with the correct class (Supported, Refutes or NotEnoughInfo), and also return the sentence(s) forming the necessary evidence for the assigned label (from Wikipedia). Out of 23 competing teams, 19 teams scored higher than the previously published baseline.

(Nie et al. 2019) set out to improve upon the previous state of the art which used an SNLI (Stanford Natural Language Inference) model. Instead of using the NLI model, they used three Neural Semantic Matching Networks (NSMN). They selected documents based on matched keywords between the title and claim and ranking the selected documents giving dis-ambiguous documents a higher score. Pass the ranked documents and compare a combination of a title and the first line with the claim using the NSMN. For sentence selection, they passed the sentences of the selected documents through an NSMN model to classify if the statements are relevant and ignoring sentences that score below a certain threshold. For RTE, they created evidence by concatenating all the sentences and comparing them to the claim using the NSMN network with a combination of a 30-dimension WordNet, 5 dimensions embedding to create unique IDs, and the scores produced by previous layers.

(Vlachos, Riedel 2014) extracted noun phrases using AllenNLP parser and fetched documents using the MediaWiki API by searching on titles against the stemmed claim. For sentence selection, a LSTM model is trained using Hinge loss and negative sampling to select sentences. For RTE, the sentences and claim are combined and then apply enhanced LSTM to each pair and then used average and max pooling and then feed the result to a multi-layer perceptron for classification.

(Malon. 2018) used a high precision entailment classifier based using transformer networks pre-trained by OpenAI. One of the caveats of this implementation is to ignore instances that required multiple statements to resolve a claim. For document retrieval and sentence selection, A combination of TF-IDF, named entities in titles, disambiguation, and capitalized expressions occurring in the first fifty lines of the document are used to get the most relevant articles. For RTE, used a custom version of OpenAI's pre-trained transformer network which was trained on the Fever Dataset.

3 Dataset

The dataset built for this challenge consists of claims that have already been labeled by human annotators. The dataset is split into ~79% training data, ~5% development data, ~5% test data and ~11% held out data. The Table 1, which was taken from the FEVER challenge paper (Thorne et al, 2018), gives the exact figures. Figure 1 explains the type of data present in the data set.

Split	Supported	Refuted	NEI
Training	80035	29775	35639
Dev	3333	3333	3333
Test	3333	3333	3333
Reserved	6666	6666	6666

Table 1: Dataset split sizes for
SUPPORTED, REFUTED and NOTENOUGHINFOR (NEI) classes

Training Data	
Id	150448
Verifiable	VERIFIABLE
Label	SUPPORTS
Claim	The 1991 NBA Finals had the Lakers in it.
Evidence	[[36004, 43545, "1991_NBA_Finals", 8]]
	[[36004, 43546, "1991_NBA_Finals", 14]]
	[[36004, 43547, "1991_NBA_Finals", 17]]
Unlabelled Data	
Id	194462
Claim	Tilda Swinton is a vegan.

Wikipedia Evidence	
id	1958–59_NBA_season
text	The 1958 – 59 NBA Season was the 13th season of the National Basketball Association . The season ended with the Boston Celtics winning the NBA Championship -LRB- the first of what would be 8 straight -RRB- , beating the Minneapolis Lakers 4 games to 0 in the NBA Finals.
lines	The 1958 – 59 NBA Season was the 13th season of the National Basketball Association. National Basketball Association National Basketball Association Basketball Basketball
	The season ended with the Boston Celtics winning the NBA Championship -LRB- the first of what would be 8 straight -RRB- , beating the Minneapolis Lakers 4 games to 0 in the NBA Finals. Boston Celtics Boston Celtics Minneapolis Lakers Minneapolis Lakers NBA Finals 1959 NBA Finals

Figure 1: Single representation of data in FEVER dataset

4 Task Definition

The FEVER challenge involves three tasks: Document Retrieval, Sentence Selection, and Recognizing Textual Entailment.

4.1 Document Retrieval

Document retrieval is defined as the matching of a user stated query against a set of free-text records (Wikipedia Document Retrieval). The documents used are a dump of Wikipedia articles from 2017. A set of relevant documents are selected and passed on to the Sentence Selection module.

4.2 Sentence Selection

Sentence selection is the task of identifying sentences that contain the answer to a given question or claim (Sentence Selection). The objective of this module is to get the similarity between a claim and a sentence. In order to verify a claim, the project aims at selecting specific sentences from the retrieved documents to support or refute the claim. A set of 5 sentences are selected and passed on to the Recognizing Textual Entailment module.

4.3 Recognizing Textual Entailment

Recognizing Textual Entailment or RTE is a generic task of capturing major semantic inference needs in Natural Language Processing applications. This means that given two fragments of text, recognize whether the meaning of one fragment of text is entailed (can be inferred) from another text (ACLWeb RTE).

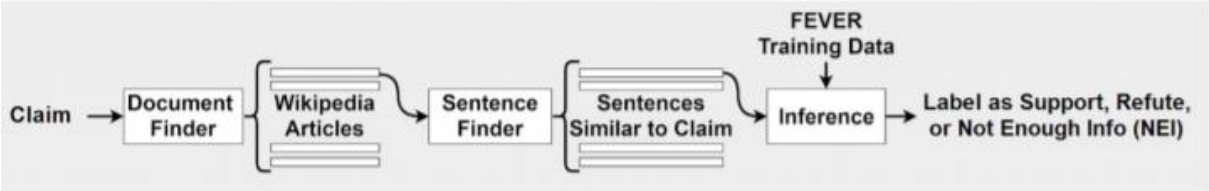

Figure 2: Visual representation of Fact Checking and Verification System

5 Baseline Implementation

A simple pipeline consisting of the three modules explained above is implemented. Each module is trained on training dataset (~180k claims), evaluated individually on the dev dataset (~ 20k claims), and the overall system is evaluated on the test (~20k claims) dataset. The baseline system is heavily inspired from the FEVER baseline (Thorne et al. 2018).

5.1 Document Retrieval

Facebook's DrQA (Chen et al. 2017) document retrieval is used to create a simple inverted index on the Wikipedia dump. Using the inverted index, TF-IDF scores are calculated for each claim and the top 5 documents are selected.

5.2 Sentence Selection

Using Facebook's DrQA (Chen et al. 2017) we extract the top 5-most similar sentences from the 5-most relevant documents using TF-IDF similarity. For TF-IDF, PyTorch implementation of DrQA is used. First, we rank sentences based on TF-IDF similarity to the claim. Then first sort most similar sentences and tune a cut-off using validation similarity on the development set. Finally, we get top results using cosine similarity based on the above scores.

5.3 Recognizing Textual Entailment

In order to perform RTE, the baseline approach is to use a Multi-Layer Perceptron (MLP) with a single hidden layer and which uses term frequencies and TF-IDF cosine similarity between the claim and evidence as features.

6 Proposed System and Implementation

A pipeline similar to the baseline system was developed for the task. As before, each module is evaluated individually on the dev dataset and the overall system is evaluated on the test dataset.

6.1 Document Retrieval

Similar to the baseline approach, we use Named Entity Recognition and Noun Phrase Extraction on each document and the results are stored in MongoDB.

Named Entity Recognition (NER) is the task of identifying named entities in unstructured documents into predefined categories. The categories used here are: ORG (organization), POI (Person of Interest), WORK OF ART (movies, books, shows, etc), DATE, EVENT (events such as NBA finals), PERSON (Emma Watson etc), and OTHER (countries, nationalities, money, cardinal numbers).

Noun Phrase Extraction is the task of identifying part of speech patterns which include a noun. spaCy was used to perform both tasks. spaCy (Honnibal et al. 2017) is an industrial strength NLP tool which has more extensive trained models compared to DrQA. Additionally, spaCy models can perform the above tasks in a vectorized manner leading to faster and better results. spaCy has pretrained models and we make use of en_core_web_sm and en_core_web_lg which are English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl.

Custom Entity Tagger models were unable to pick up a lot of movies, books, and other work of arts. As a work around, spaCy allows users to add patterns and train on top of the existing models. For this we made use of Wikipedia Disambiguation to find patterns and also mark certain entities as POI (person of interest). For example, the original model was unable to find any entities in 'True Story' (The B.G.'z album). However, using disambiguation, we can see that 'True Story' is related to 'The B.G.'z album'. So 'True Story' becomes a 'Work of Art'. Furthermore, we can see that 'The B.G'z is a band' which is also not marked and mark it as a Person of Interest. These custom tags helped improve the precision significantly.

Once the models were trained, we found entities for each claim and matched them to find the top documents and then were passed on to the next module.

6.2 Sentence Selection

Using TF-IDF and cosine similarity is a dominant method to evaluate semantic textual similarity. However, this method is limited by it's term-specificity (Mihalcea et al. 2006) and hence we experiment using a Siamese Network (Mueller et al. 2016) and biLSTM (Huang et al. 2015) model to detect similarity between a claim and a sentence. A Siamese Network, also known as twin neural network is an ANN that uses same weights while working in tandem on two different input vectors to compute comparable output vectors (Siamese Network). Siamese network can work to compare similar instances in different type sets. This network can have two or more identical subnetwork components. The weights of the network are shared and works on two different input vectors to compute comparable output vectors. Additionally, a basic LSTM model can also give rich semantic learned models and hence we use a biLSTM model which exploits information from both left and right.

Model Architecture: Figure 3 describes the model architecture of sentence selection. Pairs of claim and sentences are formulated. This is done by extracting all sentence from free-text records (Wikipedia Document Retrieval) for each predicted document (by the previous model) corresponding to the claim. Each claim and sentence are first tokenized using spaCy and then both are vectorized using GloVe embeddings. These embeddings are fed to a single layer biLSTM-Siamese network. The loss is calculated using Cross Entropy and the model is trained to make predictions. The probability of each prediction/similarity-score being true is calculated using SoftMax function. The range of probability is between 0 and 1. For each claim, all the calculated probabilities are sorted in decreasing order and top sentences with highest probability are selected. This is then fed to the next module (RTE) for inference and claim-labeling.

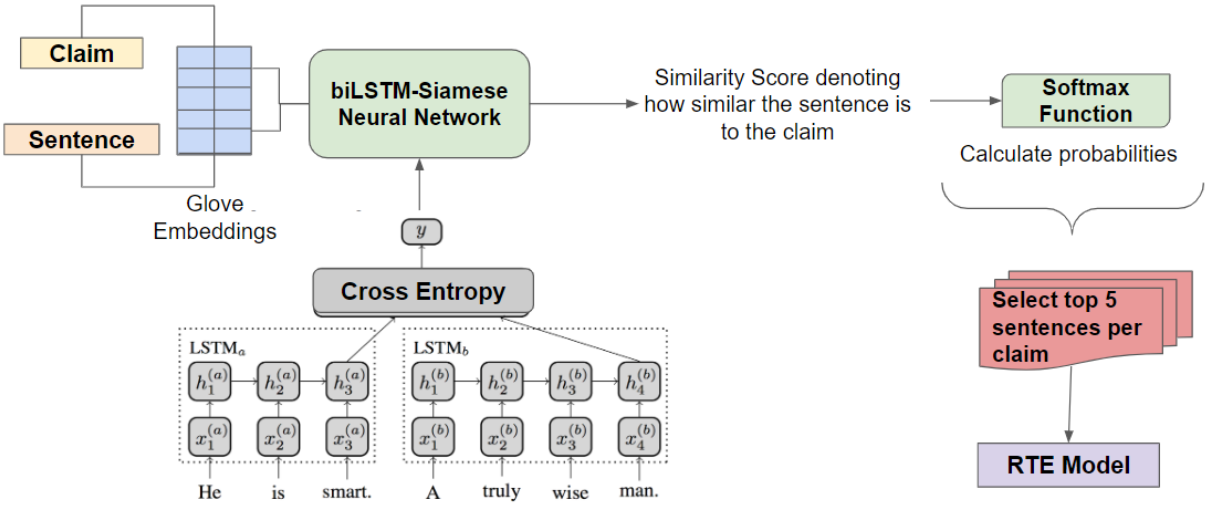

Figure 3: Sentence selection model architecture

Training: Before training the model, we pre-process the data. Firstly, we consider only verifiable data. This is because, in the training set, 'Non-Verifiable' claims by default have 'Not Enough Info' label, which means that there is no evidence set corresponding to that claim and hence no sentences can be extracted from the free-text records (Wikipedia Document Retrieval). Sentence cleaning is done, by removing hyper-links, emojis, unwanted text characters, conversion to Unicode etc. Once, sentences are cleaned, we make pairs of each sentence in the extracted article with the claim. A 'Sentence Label' is calculated by matching the line number of the sentence in the article with that of the sentence number present in the evidence set. The label is True when the values match and false otherwise. The model classifies the sentence as

The model is trained with many different valued hyperparameters. However, for this project, the following hyperparameters are finalized.

Batch Size	Epochs	Loss	Optimizer	GloVe Embeddings
128	5	Cross Entropy	Adam Optimizer	100 dimensions

Table 2: Sentence selection model's hyperparameters

6.3 Recognizing Textual Entailment

In order to predict the label of a claim, ESIM (Chen Q et al. 2017) (Enhanced Sequential Inference Model) was used, which primarily uses two bi-LSTM layers to encode the context, which is provided as the claim and the evidence sentences, and evaluates the relationship between the encoded input through a Linear layer. The input is first represented as a correspondence to vectors using the 300-dimensional GloVe embedding that was trained on a vocabulary of 840 billion words.

Model Architecture: Figure 4 below depicts the input as it flows through the first bi-LSTM layer, undergoing a series of transformations to yield an augmented representation of the intermediate output. The output then is fed to a second bi-LSTM layer, the output of which is projected onto a single vector. The path culminates with that aggregated vector being fed to a linear layer, which in turn produces a probability for each of the three labels ('Supports', 'Refutes', and 'Not Enough Info'). The output of the ESIM network is then fed to a neural aggregation network, which is a simple Multi-Layer Perceptron, which produces the final prediction on the sample. Though not shown in the diagram, Self-Attention is applied at the intermediate stage that precedes the second bi-LSTM layer, and it serves to improve the model's weight focus to better account for more salient features in the sentence. The implementation of the ESIM network was largely based on an implementation that was originally meant for a Natural Language Inference task.

Training: The model is trained end-to-end on the entire training and development datasets. The loss of the system is evaluated through Binary Cross Entropy, with the Adam optimizer to direct the gradients. For the samples for which the label was 'Not Enough Info', a preprocessing step was performed where, for each such sample, the best scoring 5 sentences from the closest 5 documents in terms of similarity to the claim were fetched using the methods described above. These 5 sentences were then

used as negative evidence to indicate to the ESIM network what resembles a case where the provided evidence is not sufficient to make a clear decision.

RTE performance is measured through the accuracy of the label classifications for a given set of claims. The evaluation on the ESIM network is carried out on each of the provided sentences, which means that each sentence is evaluated along with the claim as if it were a separate sample from the rest of the sentences in the set. Each sentence is first passed through the ESIM network, then the output of the ESIM network is then fed to a neural network, which takes the logits along with the SoftMax predictions, and outputs the single score for all the sentences in the sample.

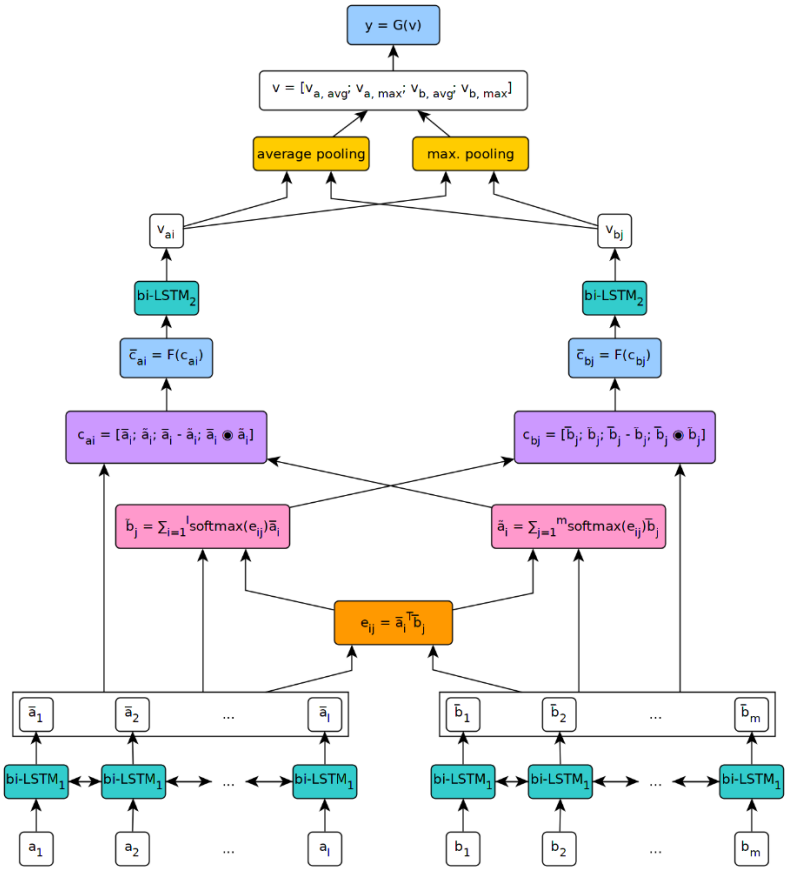

Figure 4: Block diagram of ESIM model
Source: `https://github.com/coetaur0/ESIM`

7 Results

As mentioned before, each module is evaluated individually on the dev dataset and the overall system is evaluated on the test dataset.

7.1 Document Retrieval

This module is evaluated using Macro Precision and Macro Recall as defined in (Thorne et al. 2018). The results for baseline, base spaCy model, and our custom model are listed in the table below. It is observed that the base spaCy model doesn't perform well because it missed out on quite a few movies and other works of art. This is addressed by the custom model we built and it can be seen in the increase in recall and F1.

$$\text{Macro Precision (per claim)} = \frac{\#\ docs\ correctly\ predicted}{\#\ docs\ correctly + \#\ docs\ falsely\ predicted}$$

$$\text{Macro Recall (for each claim)} = 1 \text{ if true docs} \in \text{pred docs else } 0$$

Model	Macro Precision	Macro Recall	F1
Baseline TF-IDF	19.10%	59.11%	19.67%
Base spaCy Model	9.10%	58.10%	21.60%
Custom spaCy Model	19.10%	64.90%	29.5%

Table 3: Document Retrieval Results

7.2 Sentence Selection

This module is evaluated using F1 score of fever scorer (Fever Scorer). The module is tested on original and predicted dev set. Assuming that there is 100% precision and recall for the document retrieval module, we observe, the sentence selection model gives quite descent results. However, this is an ideal scenario. In practicality, the document retrieval module cannot have 100% recall due to which the sentence selection module will not be able to extract the correct sentences for the claim. Moreover, if there is no document retrieved by the previous model for a particular claim, the sentence selection model cannot extract the article from the free-text records (Wikipedia Document Retrieval). Hence the predicted dev set (I), showcases very poor results. When the document retrieval module predicts some document even for 'Not Enough Info' claims, the sentence selection model displays improved results (II). Furthermore, if the number of predicted documents are increased from 5 to 10, we see some more slight improvements in the values (III).

Input	Precision	Recall	F1
Original dev set	29.35%	77.09%	42.51%
Predicted dev set (I)	2.02%	6.27%	3.06%
Predicted dev set (II)	7.87%	30.19%	12.48%
Predicted dev set (III)	10.56%	24.04%	10.05%

Table 4: Sentence Selection Results

7.3 Recognizing Textual Entailment

The RTE system is evaluated by considering the true labels for each of the samples in the dataset, and the overall accuracy is calculated thusly. Figure 5 shows the results that were obtained when the system was evaluated on the development and test sets. The three architectures depicted are the FEVER baseline architecture (Thorne et al. 2018), the ESIM architecture with Self Attention, with neural aggregation and without it.

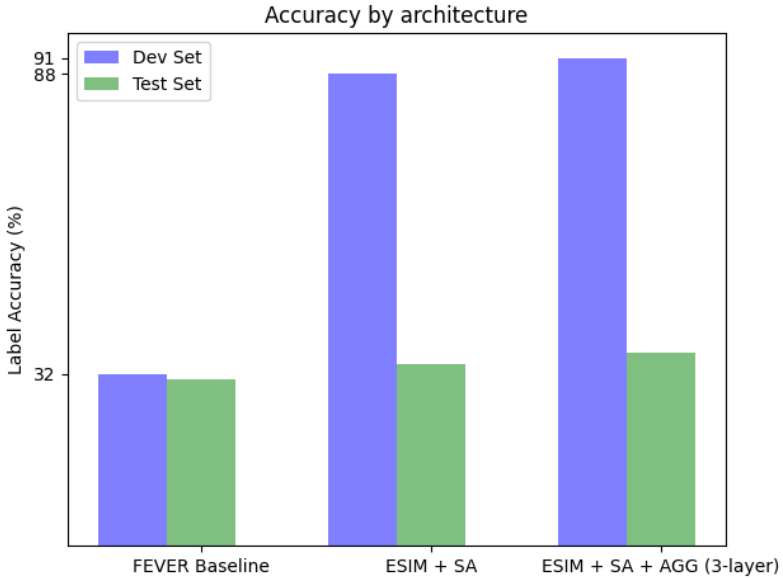

Figure 5: The accuracy of the RTE system on the training and test datasets. For each group, the left column represents the development set accuracy, and the right column represents the test set accuracy.

7.4 Overall System

The predictions for the provided test set were submitted to the official ranking system made available by the FEVER competition organizers. The results that were obtained were a slight improvement from the FEVER score of the baseline system, which was around 27%, to 34% using the ESIM network with neural aggregation. The three architectures depicted in Figure 6 are the FEVER baseline architecture (Thorne et al. 2018), the ESIM architecture with Self Attention, with neural aggregation and without it.

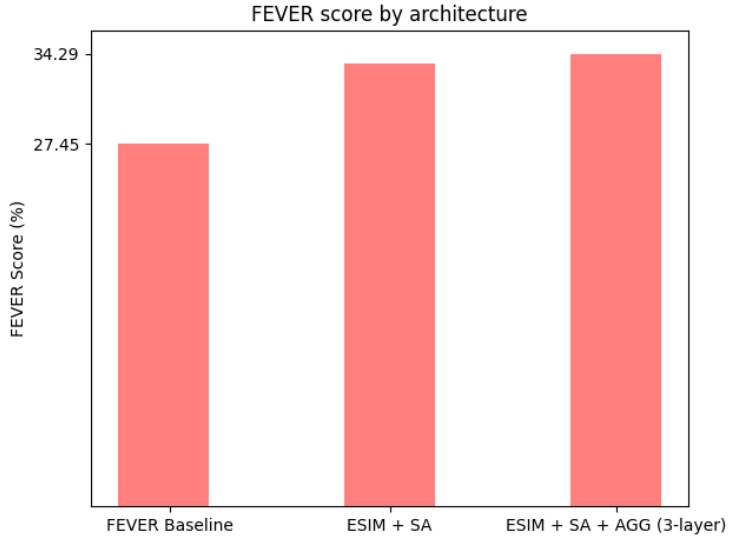

Figure 6: The FEVER scores evaluated on the test dataset for each architecture

Model	Test dataset score	Dev dataset score
Label Accuracy	41.15%	91%
Evidence Precision	1.87%	29.35%
Evidence Recall	5.66%	77.09%
Evidence F1	2.81%	42.51%

Table 5: Results

8 Conclusion

A 3-stage fact checking and claim verification model was built. We first built a baseline approach and then experimented various concept in Natural Language Processing and Text Mining. Our system was able to achieve a FEVER Score of 34.29% surpassing the baseline (27.45%) model. However, at 64.21%, the state-of-the-art model, motivates us to improve our model.

9 Future Improvements

Keeping in mind the time allotted for this task and the strength of the team, we believe that a decent model was built and the objective and set goals were successfully achieved. However, like all models, our model can further be improved by using different embeddings – ELMo, fastText. In the current sentence selection model, 100 dimensions GloVe embeddings are used and hence 300 dimensions embeddings can certainly improve the precision. Deeper models such as BERT can be further explored and NLP concepts such as model ensembling and annealed sampling can be implemented. One can focus on improving recall in the document retrieval module and precision in sentence selection module in order to have higher accuracies during recognizing textual entailment.

All the model weights and data can be found on `https://github.com/sridhara-nirudh/tmpFever`

10 Participation

The work was roughly divided such that each team member worked on one of the three modules in the system. Concretely, the work was carried out as follows:

Team Member	Task
Anirudh Sridhar	Document Retrieval
Snigdha Gupta	Sentence Selection
Anas Ahamed	Recognizing Text Entailment
Anas Ahamed	Frontend application

Table 6: Team Participation

Acknowledgements

Credits We would like to extend our gratitude to our professor, Dr. Rohini K. Srihari for giving us an opportunity to work on this project. We would also like to thank our teaching assistant Archita Pathak for helping us in completing this project and guiding us whenever needed. We thank FEVER Workshop Organizing Committee for organizing the FEVER Shared Task Challenge and giving us a chance to learn from the works of previous teams. Through this project we were able to learn concepts of Natural Language Processing and Text Mining and implement them practically allowing us to learn and continue our interests in the field.

Reference

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA. Association for Computational Linguistics.

Chen Q, Zhu X, Ling Z, Wei S, Jiang H., Inkpen D. 2017. Enhanced LSTM for Natural Language Inference. arXiv:1609.06038v3 [cs.CL]

Christopher Malon. 2018. Team Papelo: Transformer networks at FEVER. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER), pages 109–113, Brussels, Belgium. Association for Computational Linguistics*.

Danqi Chen, Adam Fisch, Jason Weston, Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions

Honnibal, Matthew and Montani, Ines. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Arpit Mittal. 2018. FEVER: a large-scale dataset for Fact Extraction and VERification

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence.*

Mihalcea, R.; Corley, C.; and Strapparava, C. 2006. Corpusbased and Knowledge-based Measures of Text Semantic Similarity. *In AAAI Conference on Artificial Intelligence.*

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR, abs/1508.01991.*

Wikipedia Document Retrieval, `https://en.wikipedia.org/wiki/Document_retrieval.`

Sentence Selection, `https://www.researchgate.net/publication/269117118_Deep_Learning_for_Answer_Sentence_Selection`

ACLWeb RTE, `https://aclweb.org/aclwiki/Recognizing_Textual_Entailment.`

Siamese Network, `https://en.wikipedia.org/wiki/Siamese_neural_network`

Fever Scorer, `https://fever.ai/index.html`

CSE635 Tmp Team https://github.com/sridharanirudh/tmpFever

ESIM Model, https://github.com/coetaur0/ESIM

MaLSTM, https://medium.com/mlreview/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07

MalSTM for Text Similarity, https://medium.com/@gautam.karmakar/manhattan-lstm-model-for-text-similarity-2351f80d72f1

DeepLSTM Siamese network for Similarity https://github.com/GKarmakar/deep-siamese-text-similarity