FEVER: Fact Extraction and VERification

Jongmin Yoon
Final Project for CS224U Natural Language Understanding, Spring 2018
Stanford University
jmyoon@stanford.edu

Abstract

Verification of textual claims against textual sources is a central problem in automated fat-checking. In this paper, we build a integrated pipeline which finds related documents for a given claim and verifies it. We use Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018) which composed of 185,445 claims generated by altering sentences extracted from Wikipedia. The dataset is labeled as SUPPORTED, REFUTED, and NOTENOUGHINFO with necessary evidences for the judgenment. We employ TF-IDF and PMI approaches to the term-document frequency matrix to retrieve most relevant documents and sentences. Simple linear classification with overlapping words and word cross-product feature function results in 37% and 42% accuracy, respectively, and it is comparable to the results of the baseline approach in the original FEVER paper.

1 Introduction

With an unprecedented amount of textual information available, the demand of automating the process of fact-checking has been increasing, especially in the context of compuational journalism. Particularly, the last US presidential election in 2016 ignited the public awareness of the necessity of ordinary citizens to fact-check, as fake news and its dissemination via social media have shown an unparalled influence.

For the final project of this course, we plan to tackle a central problem in fact-checking: verification of textual claims against textual sources. Our choice of final project is motivated by the recently provided large datset Fact Extraction and VERification (FEVER) (Thorne et al., 2018). Using the FEVER dataset, we will build a system which checks whether a given textual claim is supported, refuted, or indeterminate based on a large evidence corpus. Although the original FEVER challenge requires to provide evidence sentences for the supported and refuted cases, in this project we focus simply on verification without providing evidences.

Our work is in essence an attempt to integrate two important areas in natural language understanding: Information Retrieval and Recognizing Textual Entailment. More concretely, we envision to build a system that takes a claim (e.g. a sentence) as an input and does the following tasks:

1. Document Retrieval - find most relevant documents in the database.

2. Sentence Selection - find most relevant sentences in those documents.

3. Recognizing Textual Entailment - recognize wheter the input is supported, refuted, or indeterminate by those sentences.

We would like to emphasize that our main goal is to implement basic techniques we have learned in this course and integrate them into a single pipeline. Our approach can be seen as a simplified but modified version of the baseline approach in the original FEVER paper. The entire source code for this project can be found at https://github.com/jongminyoon/fever.

2 Related Works

Since the FEVER challenge is newly created for 2018 Conference on Empirical Methods in Natural Language Processing, there has not been many published works directly tackling this data, besides the baseline results in the original FEVER paper (Thorne et al., 2018). The baseline approach implemented in the original FEVER paper is as follows. For the Task 1 Document Retrieval, it uses the document retrieval component from the DrQA system (Chen et al., 2017). It returns the k nearest documents for a query using cosine similarity between unigram and bigram TF-IDF vectors. For the Task 2 Sentence Selection, the baseline system ranks sentences by TF-IDF similarity to the claim. For the Task 3 RTE, the baseline system employs two approaches. First, it uses a multi-layer perceptron (MLP) with a single hidden layer and uses term frequencies and TF-IDF cosine similarity between claim and evidences (Riedel et al., 2017). Second, it uses a decomposable attention (DA) model between the claim and the evidence passage (Parikh et al., 2016). The second one is selected as it was the highest scroing system for the Stnaford Natural Language Inference task (Bowman et al., 2015) at the time when the baseline approach was being built.

Although the baseline paper is the only work on this datset at the moment, this task has a variety of related fields in computational linguistics. Two particular directions of research worth mentioning are Recognizing Textual Entailment (RTE) and fact-checking.

For RTE, the summary paper on the status of PASCAL RTE challenges (Dagan et al., 2009) gives a consise introduction of the motivation and importance of textual entailment research. More recently, the Stanford Natuaral Language Inference Corpus (Bowman et al., 2015), which is created from cpations of photos, has been the standard dataset of evaluating performance of RTE systems. The two biggest advantages of SNLI corpus over PASCAL RTE challenge is well explained in the SNLI paper. SNLI corpus is three order of magnitude bigger (570K pair vs less than a thousand) and this made data-driven and learning-based methods compete and surpass simple linguistic modelings. Particularly, neural network techniques, which can be only useful with a large training dataset, finally became a meaningful approach to the textual entailment research.

For fact-checking, there are not as many previous literatures as RTE, probably due to its complication and lack of large datasets. Among the few attempts on this problem are Vlachos and Riedel (2014), and Wang (2017). The two papers show difficulty of fact-checking over simple textual entailment. This challenge is even obvious from the size of available datasets; the datasets Vlachos and Reidel and Wang used in their research are much smaller than that of SNLI (106 and 13k vs 570k in SNLI). Also, the full-fledged automated fact-checking requires more tasks than simple textual entailment, particularly in extracting supporting evidences from a bigger textual source and considering the context of the claim. The study by Wang shows some attempts to include those contextual metadata in his analysis.

The FEVER challenge provides an important middle step between textual entailment and full fact-checking. It requires not only making a verdict on the truthfulness of the claim but also providing supporting evidences of the verdict. Therefore, its scope is beyond the simple textual entailment of RTE and SNLI. However, it is only based on the factual descriptions in Wikipedia and therefore we do not need to consider context information yet. With this restriction, the authors of FEVER challenge were able to make a dataset of size 185k from Wikipedia source. With the large size of the dataset, we can try learning-based methods more significantly.

3 FEVER Dataset

The FEVER dataset was constructed in two stages: claim generation and claim labeling. The background corpus is June 2017 Wikipedia dump, processed with Stanford CoreNLP and sampled sentences from the introductory sections of approximately 50,000 popular pages. The annotators generated 185,445 claims by altering sentences in the introductory section. In the claim labeling stage, separate annotators, who did not know how the claims are generated, classified them into three classes, SUPPORTED, REFUTED, and NOTENOUGHINFO. For SUPPORTED and REFUTED classes, the annotators also provided evidence sentences which support or refute the claim, in terms of Wikipedia URL and sentence number. Note that often there are several sentences which are simultaneously required to make a verdict. The labels are checked partially by authors and expert annotators to check its validity. The details of claim generation and labeling can be found in the original FEVER paper (Thorne et al., 2018). Table 1 shows how many claims are for each class. Note that the reserved split is for the EMNLP 2018 competition.

4 Our Approach

From the provided pre-processed Wikipedia dump, we first build a SQLite database. The total number of documents is 5,416,537. With the database file, we begin with the document retrieval part of the pipeline.

4.1 Document Retrieval

We build the term-document frequency matrix from the database. We use NLTK sentence tokenizer to identify sentences, and then apply regular expression tokenizer using ([\w-]{3,}) so that we extract word characters and dash symbol with three or more characters only. After then, we use Porter stemmer to stem each word. Finally, given the size of large corpus, we decide to use hash function to represent words, following the original FEVER paper. We use unsigned 32 bit murmurhash. After all of this processing, we build a unigram term-count frequency matrix.

It should be noted that creating the frequency matrix at once from the entire database is infeasible with a commercial personal laptop. Therefore, we divide the database into five files, build a frequncy matrix for each file and then merge the matrices to get the full term-document frequency matrix. After then, we use two schemes for reweighting, TF-IDF and PMI. For term frequency tf, document frequency df, total count t, the precise equations used are

$$\text{tfidf} = \log(tf + 1) \times \log \frac{t - df + 0.5}{df + 0.5}$$

$$\text{pmi} = \log(tf + 1) \times \frac{t}{\text{colsum} \cdot \text{rowsum}}$$

The choice of $\log(1+tf)$ for both schemes is from computational motivation, as it is computationally much faster than actually normalizing the term frequency across all of the 5.4 million documents. From the TF-IDF and PMI matrices, we find closest documents. We vectorize the input claim using word frequency and log(1+termfreqs), and then dot-product it with the term-document matrices. We choose n documents with highest scores as the closest documents to the input claim.

4.2 Sentence Selection

From the chosen n documents, we select most relevant senteces to the input claim. We first recognize all sentences of those n documents and vectorize each sentence using word frequency and log(1+termfreqs). Simple dot product with the vectorized input claim gives a score to determine how similar each sentence is to the given input claim. We choose m closest sentences.

4.3 Sampling for NotEnoughInfo

It must be noted that we need to sample sentences for the NOTENOUGHINFO class. For SUPPORTED and REFUTED classes, the annotators have provided evidence sentences based on which we can make a verdict. Therefore, for those two classes we can directly use those sentences as input for our feature function for training. However, NOTENOUGHINFO class does not have evidence sentences and therefore, we sample sentences for NOTENOUGHINFO class. We utilize the document retrieval and sentence selection part described above; we select the m closest sentences from n closest documents as training data for NOTENOUGHINFO class.

4.4 Recognizing Textual Entailment

For RTE, we use two simple feature functions, overlapping words and word cross-product. We use multinomial logistic regression classifier with grid cross-validation for hyperparameters.

Split	Supported	Refuted	NotEnoughInfo
Training	80,035	29,775	35,639
Dev	3,333	3,333	3,333
Test	3,333	3,333	3,333
Reserved	6,666	6,666	6,666

Table 1: Dataset split sizes for Supported, Refuted, and NotEnoughInfo classes.

	Oracle Accuracy (%)	
Num Docs	TF-IDF	PMI
1	23.2	22.1
3	45.5	46.2
5	56.9	56.6
10	69.0	68.9

Table 2: The oracle accuray in percent for two reweighting schemes and number of documents retrieved.

5 Result

5.1 Document Retrieval

For each claim in the training set with Supported or Refuted label, where evidences are given, we compare the gold evidence documents with what our document retrieval system(TF-IDF or PMI with dot product) gives as most relevant. Table 2 shows the oracle accuracy for our document retrieval system to include gold evidence document. As expected, the oracle accuracy increases as we increase the number of documents we retrieve. The oracle accuary in our test shows good agreement with the baseline paper (25%, 55%, 66% for $n = 1, 5, 10$). From this observation, we choose $n = 5$ for training data samping for NotEnoughInfo class for sufficient oracle accuracy and efficiency.

5.2 Sentence Selection

We compare the gold evidence sentences with what our sentence selection system gives as most relevant. To make independent and separate test on sentence selection from that of the document retrieval, we select sentences from the gold evidence documents and compare them with the gold evidence sentences, instead of using our document retrieval system to provide base documents for our sentence selection. Table 3 shows the oracle accuracy for our sentence selection system. Again, as we increase the number of sentences retrieved, the oracle accuracy increases. For sentence selection part, we choose $m = 5$ for training data samping of NotEnoughInfo class for sufficient oracle accuracy and efficiency.

Num Sentences	Oracle Accuracy (%)
1	51.2
3	67.0
5	72.7
10	81.8

Table 3: The oracle accuray in percent for number of sentences retrieved.

	Precision	Recall	F1 score
Supported	0.337	0.798	0.455
Refuted	0.426	0.012	0.023
NEI	0.362	0.326	0.343
avg / total	0.374	0.346	0.274

Table 4: Precision, Recall, and F1 score for overlapping words feature function. NEI for NotEnoughInfo.

5.3 Recognizing Textual Entailment

The textual entailment part is a 3-class classification problem, so a random system will give about 33% accuracy. We evaluate classification accuracy, recall, and the following F1 score with two types of feature function, 1) overlapping words and 2) word cross-product. We use the multinomial logistic regresssion classifier and grid-search for best hyperparameters. Our grid-search includes $L1$ and $L2$ for penalty function and {0.4, 0.6, 0.8, 1.0} for inverse regularization strength. Table 4 shows the result for overlapping-words feature function. The best result was obtained with $L2$ penalty and regularization strength 1.0. Table 5 shows the result for word cross-product feature function. The best result was obtained with $L1$ penalty and regularization strength 1.0.

	Precision	Recall	F1 score
SUPPORTED	0.378	0.410	0.394
REFUTED	0.535	0.219	0.311
NEI	0.339	0.527	0.420
avg / total	0.421	0.385	0.375

Table 5: Precision, Recall, and F1 score for word cross-product feature function. NEI for NOTENOUGHINFO.

6 Analysis

For overlapping-words feature function, we can see the accuracy is only slightly better than what would have been obtained from the random guess. However, it should be noted that even in the baseline approach in the original FEVER paper (Thorne et al., 2018), the accuracy was 40% and 50%, for Multi-Layer Perceptron approach and Decomposable Attention model, respectively. This shows the challenging nature of this task.

Although the baseline approach shows better precision, it only reports the accuracy without recall or F1 score, so there is a limitation in comparing the two results. In our apporoach using the overlapping words feature function, it should be noted that the accuracy for REFUTED class is relatively higher than the other two classes, but its recall is very low. The SUPPORTED class shows high recall.

The word cross-product feature function shows overall winning over the overlapping words feature function, but compensates some of advantages of the overlapping word approach. The average precision increased to 0.421 from 0.374. The recall for REFUTED class shows a big improvement, but the recall for SUPPORTED class drops. Also it is notable to see that this simple approach of word cross-product gives 42.1% average accuracy, which is comparable or slightly better than the MLP approach which was chosen in the baseline paper as well-performing among linear classification methods. Given the fact that the DA method which requires large computation time for training shows 52% accuracy, we can see that word cross-product approach is very robust.

7 Lessons

Although the each step and method we used in this project is basic and all covered in the lectures, we find that integrating those elements into a single pipeline gave us another depth of understanding how each ingredient works. Working on this project provided us an experience of building a system which performs the enitre stack of operations from the very beginning of data preprocessing to classification learning. It was good to see how all the ingredients - tokenizers, stemmer, unigram, hash function, and reweighting schemes, textual entailment training - are interwoven together to produce a system which does an actual work.

Another lesson we had was the issue of scalability. Particularly, as we deal with 5.4 million documents from the Wikipedia, there were a few tricky issues in handling the database because of its large size. This enforces us to think how to divide & conquer the data so that our approach can be feasible using a personal laptop, and also some of parallel programming techniques in Python. We had to use sparse matrces in SciPy package rather than NumPy arrays, for large sparse matrix, we indeed was able to see that a certain order of matrix multiplication is much faster than the other, etc. This also restrained our choice of reweighting and vectorization scheme, particularly in using $\log(1 + tf)$ rather than actual normalization.

We hope that we had had more time to work on this project and improve. Latent Semantic Analysis on the TF-IDF or PMI matrices could have increased the document retrieval and sentence selection accuracies. For making a verdict on the truthfulness of the claim, we could have use query expansion using the similar words using GloVe vectors. Also, we could have tried vectorized feature functions using GloVe vectors too. It would be interesting how these attempts can improve the system.

Acknowledgments

This work is a final project for CS 224U Natural Language Understanding, Spring 2018 at Stanford University.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv:1508.05326 [cs]*. ArXiv: 1508.05326.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *arXiv:1707.03264 [cs]*. ArXiv: 1707.03264.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for Fact Extraction and VERification. *arXiv:1803.05355 [cs]*. ArXiv: 1803.05355.

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.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.