AttentiveChecker: A Bi-Directional Attention Flow Mechanism for Fact Verification

T.Y.S.S.Santosh, Vishal G, Avirup Saha, Niloy Ganguly
Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
India
{santoshtyss, arsrivish2, saha.avirup}@gmail.com ,
niloy@cse.iitkgp.ac.in

Abstract

The recently released FEVER dataset provided benchmark results on a fact-checking task in which given a factual claim, the system must extract textual evidence (sets of sentences from Wikipedia pages) that support or refute the claim. In this paper, we present a completely task-agnostic pipelined system, AttentiveChecker, consisting of three homogeneous Bi-Directional Attention Flow (BIDAF) networks, which are multi-layer hierarchical networks that represent the context at different levels of granularity. We are the first to apply to this task a bi-directional attention flow mechanism to obtain a query-aware context representation without early summarization. AttentiveChecker can be used to perform document retrieval, sentence selection, and claim verification. Experiments on the FEVER dataset indicate that AttentiveChecker is able to achieve the state-of-the-art results on the FEVER test set.

1 Introduction

The rising influence of fake news poses a clear threat to ethical journalism and the future of democracy. In order to tackle the sheer volume of fake news produced, robust automatic techniques to counter it need to be developed. To that end, in order to facilitate researchers to develop algorithms, a number of fact checking datasets have been released in the recent past (Vlachos and Riedel, 2014), (Wang, 2017), (Ferreira and Vlachos, 2016), (Pérez-Rosas et al., 2017), the 2017 Fake News Challenge (Pomerleau and Rao, 2017) dataset, the dataset released against Triple Scoring Task at the WSDM Cup 2017 (Heindorf et al., 2017) etc.

However, none of these datasets provide manual annotation for sentence or phrase-level evidence. In this work, we experiment on the Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018a), which is one of the first which provides sentence-level annotations. (An Arabic corpus (Baly et al., 2018) has been recently released.) A shared task corresponding to the dataset was floated that required verification of an input claim with potential evidence in a large database of about 5 million Wikipedia documents, and also provided a standardized benchmark setting which enabled easy and fair comparison. Table 1 lists the dataset splits and sizes.

Split	Supports	Refutes	NotEnough Info
train	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: Statistics of claims in FEVER

Several attempts have been made to tackle the task defined by FEVER, the most notable ones being (Nie et al., 2018; Yoneda et al., 2018; Hanselowski et al., 2018) which secured the 1st, 2nd and 3rd place on this shared task respectively (Thorne et al., 2018b). Diverse methods were applied, mostly using task-specific features which allowed them to beat the baseline given by (Thorne et al., 2018a). In this paper, we propose a completely task-agnostic system, AttentiveChecker to tackle the FEVER task.

AttentiveChecker is a pipelined system consisting of three identical Bi-Directional Attention Flow (BIDAF) networks, which are multi-layer hierarchical networks that represent the context at different levels of granularity. We use a bi-directional attention flow mechanism where we allow the context vector at each step to flow to the next layers in the BIDAF model. This helps to obtain a query-aware context representation without

Proceedings of NAACL-HLT 2019, pages 2218–2222
Minneapolis, Minnesota, June 2 - June 7, 2019.

early summarization. This is different from previously used attention layers employed in (Sordoni et al., 2016), (Shen et al., 2017) where the query and context are summarized into a single feature vector. AttentiveChecker achieves a FEVER Score of 66.72 on the test set, which beats the 1st ranked system (Nie et al., 2018) by more than 2 points.

2 Task Definition

The task can be described as verifying a claim using evidence from Wikipedia. The system must label the claim as SUPPORTED or REFUTED based on the evidence from Wikipedia or NotEnoughInfo if there is not sufficient evidence to either support or refute it. The system must also extract textual evidence (a set of sentences from Wikipedia pages) that support or refute the claim. A prediction is said to be correct only if both (a). the label is correct and (b). the predicted evidence set (containing at most five sentences) covers the annotated evidence set. The accuracy in percentage of such predictions is called the FEVER score. The overall task can be compartmentalized into three distinct subtasks: (i). identifying relevant documents from Wikipedia (Document Retrieval), (ii). selecting sentences forming the evidence from the documents (Sentence Selection) and (iii). classifying the claim w.r.t. collected evidence (Claim Verification).

2.1 Document Retrieval

For this sub-task, we provide the first sentence of the document and the claim as the two input sequences to the BIDAF model which outputs the probability of selecting the current document as evidence. As the number of documents is huge, we first reduce the search space by performing keyword match with titles of Wikipedia pages i.e. the document is selected if there is an exact match between the title and a span of text in the input claim. We use our BIDAF model to rank the chosen documents in order of relevance. The top-k documents based on their score are shortlisted for the next level.

2.2 Sentence Selection

For this sub-task, we provide (i). each sentence of the documents in the evidence set and (ii). the claim as the two input sequences to the BIDAF model which outputs the probability of selecting the current sentence as an evidential sentence. Since the search space is already reduced by Document Retrieval, we can directly traverse all the sentences and compare them with the claim using the BIDAF model. We rank all the sentences in every document from the evidence set and choose the top-k sentences.

2.3 Claim Verification

For this sub-task, we provide (i) the claim, and (ii) all evidential sentences together with the corresponding document names (to address coreference issues) as the two input sequences to our BIDAF model. The BIDAF model outputs the scores for three labels, namely SUPPORTED, REFUTED and NotEnoughInfo. Then the claim is labelled as the one with the highest score. Note that in order to have fair comparison across methods, the FEVER challenge limits the sentences in the evidence to a maximum of five. Also in the test set of FEVER data, the number of sentences providing evidence is at most five.

3 The BIDAF Model

In this section, we will describe the architecture of the Bi-Directional Attention Flow (BIDAF) model which constitutes the basic building block of AttentiveChecker. In each stage of the pipeline, the BIDAF model takes two input sequences and outputs labels based on the particular sub-task being performed. Let X and Y denote two input word sequences of length m and n respectively. The BIDAF model consists of four layers: (i). the embedding layer takes raw text sequences X and Y as input and encodes them into suitable vector sequences $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$, (ii). the attention layer takes $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ as input, computes the attention scores of each sequence w.r.t. the other, and outputs two attended sequences $\mathbf{C}$ and $\mathbf{D}$, (iii). the modeling layer takes $\mathbf{C}$ and $\mathbf{D}$ as input and outputs two fixed size vectors $\hat{C}$ and $\hat{D}$ which capture the semantic similarity between the two sequences, and (iv). the output layer takes $\hat{C}$ and $\hat{D}$ as input and provides the scores for the output labels. The layers are described below in detail.

Embedding Layer: In the embedding layer the input sequences are encoded at three levels of granularity viz. character, word and context. We obtain the character-level embedding of each word using Convolutional Neural Networks (CNN) as described in (Kim, 2014). For word level en-

coding, we use pre-trained word vectors, to obtain the word embedding of each word in the input sequences. Corresponding to each word, we output a vector concatenating the word level and character level encoding. Let $\mathbf{X'}$ denote the sequence of concatenated vectors for the input word sequence X. We use a Bi-directional Long Short-Term Memory Network (BiLSTM) on $\mathbf{X'}$, to model the temporal interactions between words within each sequence and thus obtain the contextual embedding :

$$\hat{\mathbf{A}} = \mathbf{BiLSTM}(\mathbf{X'}) \in \mathbb{R}^{d_0 \times m}$$

where $\hat{\mathbf{A}}$ denotes the sequence of all output vectors of the BiLSTM. Similarly we get a sequence $\hat{\mathbf{B}} \in \mathbb{R}^{d_0 \times n}$ for the sequence Y. Note that the character-level embeddings obviate the need for task-specific embeddings or features (e.g. those used by (Nie et al., 2018) for claim verification), as character-level embeddings can encode a far more generalized set like numeric sequences, misspellings, emoticons or other languages.

Attention Layer: Intuitively, the attention layer give higher importance or weight to those parts of a sequence which overlap with parts of the other sequence. We compute attention for the two sequences $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ with respect to each other. Here we compute attention in both directions: from first sequence to second sequence and vice versa. To achieve this we make use of a similarity matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ where $\mathbf{S_{ij}}$ indicates the similarity (or attention score) between the i^{th} word in the first sequence and the j^{th} word in the second sequence and is computed by applying a linear mapping after a single layer perceptron stage on the i^{th} column vector of the first sequence and the j^{th} column vector of the second sequence (Hermann et al., 2015).

$$\mathbf{S_{ij}} = \mathbf{W_1}^{\intercal}.\tanh(\mathbf{W_2}.[\hat{\mathbf{A}}_i : \hat{\mathbf{B}}_j] + \mathbf{b}) \in \mathbb{R}$$

where $\mathbf{W_1}$, $\mathbf{W_2}$ indicate trainable weight matrices, $\mathbf{b}$ indicates trainable bias matrix, $\hat{\mathbf{A}}_i$, $\hat{\mathbf{B}}_j$ indicate i^{th} column vector of $\hat{\mathbf{A}}$ and j^{th} column vector of $\hat{\mathbf{B}}$ respectively. : indicates vector concatenation. The context vector for the i^{th} word of the sequence $\hat{\mathbf{A}}$ w.r.t. the sequence $\hat{\mathbf{B}}$ is given by

$$\tilde{\mathbf{A}}_i = \sum_j \alpha_{ij} \hat{\mathbf{B}}_j \in \mathbb{R}^{d_0}$$

where $\alpha_{ij} = \text{softmax}_j(\mathbf{S_{ij}}) = \frac{\exp(\mathbf{S_{ij}})}{\sum_{j=1}^{n} \exp(\mathbf{S_{ij}})}$.

Finally we obtain the attention vector sequence for the sequence X as $\mathbf{C} \in \mathbb{R}^{d_1 \times m}$ by adding ReLU after applying single layer perceptron to the vector obtained by concatenating the i^{th} column vector of contextual embedding ($\hat{\mathbf{A}}$) and i^{th} column vector of context vectors ($\tilde{\mathbf{A}}$):

$$\mathbf{C}_i = \max(0, \mathbf{W}.[\hat{\mathbf{A}}_i : \tilde{\mathbf{A}}_i] + \mathbf{b}) \in \mathbb{R}^{d_1}$$

where $\mathbf{C}_i$, $\mathbf{W}$, $\mathbf{b}$ indicate i^{th} column vector of $\mathbf{C}$ corresponding to the i^{th} word of X, trainable weight matrix, trainable bias matrix, respectively; : indicates concatenation of vectors. Each column vector of $\mathbf{C}$ can be considered as the Y-aware representation of each word in the sequence X. The above computation is repeated to obtain the context vector $\tilde{\mathbf{B}}$ and subsequently the attention vector sequence $\mathbf{D} \in \mathbb{R}^{d_1 \times n}$ for Y.

Modeling Layer: We apply bi-directional LSTM to the obtained sequence $\mathbf{C}$ for X to obtain a new sequence $\tilde{\mathbf{C}}$

$$\tilde{\mathbf{C}} = \mathbf{BiLSTM}(\mathbf{C}) \in \mathbb{R}^{d_2 \times m}$$

and then take the concatenation of the final forward and backward outputs of the BiLSTM to obtain a fixed size vector representation $\hat{C} = \overleftarrow{\tilde{\mathbf{C}}}_1 : \overrightarrow{\tilde{\mathbf{C}}}_m \in \mathbb{R}^{2d_2}$ which captures the semantic interaction between the two sequences because of the attention applied in the previous layer. This is different from the embedding layer, which captures the interaction among words of one sequence independent of the other sequence. Similarly we get a sequence $\tilde{\mathbf{D}} \in \mathbb{R}^{d_2 \times n}$ and a vector $\hat{D} = \overleftarrow{\tilde{\mathbf{D}}}_1 : \overrightarrow{\tilde{\mathbf{D}}}_n \in \mathbb{R}^{2d_2}$ for Y.

Output Layer: To quantify the semantic similarity between the two sequences, we apply the inverse exponential of the Manhattan distance (M) as suggested in (Mueller and Thyagarajan, 2016) to the representations obtained from the modeling layer.

$$\mathbf{O} = M(\hat{C}, \hat{D}) = \exp(-||\hat{C} - \hat{D}||_1)$$

Then $\mathbf{O}$ is fed to a single layer perceptron, to obtain the required sub-task specific output. For document retrieval and sentence selection, we have one output neuron in the single layer perceptron which indicates the probability of selecting the current document or sentence as evidence, while for claim verification, we have three output neurons indicating the scores for three labels, namely SUPPORTED, REFUTED and NotEnoughInfo.

We note that although the output layer must necessarily be sub-task specific (since the objective of each subtask is different), however the difference in architecture is only in the number of output neurons.

4 Results

In this section we first present the results for the full system and then the ablation results for each stage of the pipeline.

Full Pipelined system: We evaluated our complete system with all components on the test set by setting k=5 for both document retrieval and sentence selection. We found that the accuracy values obtained by AttentiveChecker (a). with the requirement to provide correct sentences as evidence (FEVER Score) and (b). without the requirement to provide correct sentences as evidence (Label Accuracy) for the SUPPORTED/REFUTED labels are 66.72 and 69.98 respectively. Table 4 compares performance of AttentiveChecker with two baselines (i). the FEVER baseline given by (Thorne et al., 2018a) and (ii). NSMN (Nie et al., 2018). We observe that AttentiveChecker performs better than the baselines on the overall task.

Task	Metric	FEVER Baseline	NSMN	Our System
Full System	Label Acc.	50.91	68.16	69.98
Full System	FEVER Score	31.87	64.23	**66.72**

Table 2: Performance of AttentiveChecker on the test set vis-a-vis the baseline systems for k=5.

NSMN (Nie et al., 2018) is a homogeneous BiLSTM-based pipeline but still uses task-specific features as input for claim verification. The key difference in AttentiveChecker compared to NSMN is the attention layer which we claim to be a better way of matching corresponding parts of the two sequences than the sequence alignment which is done in the analogous 'alignment layer' of NSMN. This attention layer is the major reason behind our improvement over NSMN and justifies the name of our system.

We now present the ablation results for the individual pipeline stages.

Document Retrieval & Sentence Selection: Table 4 shows the performance of our document retrieval and sentence selection systems on the dev set for different values of k (no. of documents/sentences retrieved). We report the Oracle Accuracy which is the upper bound of the FEVER score assuming perfect downstream stages.

k	Oracle Acc. (Document Retrieval)	Oracle Acc. (Sentence Selection)
3	93.30	82.37
5	94.05	88.23
7	94.15	88.24

Table 3: Performance of the retrieval systems on the dev set for top-k retrieval using k = 3, k = 5, and k =7

Claim Verification: To understand how well AttentiveChecker performs in this sub-task, we performed an oracle evaluation on the dev set by providing a gold standard evidence set and achieved an accuracy of 90.63 (this has to be done with k = 5 as per requirement of the shared task). The ablation results are summarized in Table 4 for k=5 and compared with the baselines. We observe that AttentiveChecker performs better in most sub-tasks (except sentence selection where it falls marginally short) compared to the baselines.

Task	Metric	FEVER Baseline	NSMN	Our System
Doc. Retrieval	Oracle Acc.	70.20	92.42	94.05
Sent. Selection	Oracle Acc.	62.81	91.19*	88.23
Claim Verif.	Oracle Acc.	88.00	N/A	90.63

Table 4: Ablation results of pipeline stages on the dev set vis-a-vis the baseline systems for k=5.
*With Annealed Sampling. Score without it is 86.65.

5 Conclusion

We developed a homogeneous BIDAF model for all three FEVER subtasks achieving the state-of-the-art on the overall task. Our system is completely task-agnostic and can therefore be transferred to other similar tasks (e.g. exaggeration detection) if need be. For the first time in this task, we have used a query-aware bi-directional attention model that avoids early summarization. Although the improvement in the FEVER Score ap-

pears modest (2 points), it is still significant considering the hardness of the problem.

References

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. *arXiv preprint arXiv:1804.08012*.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint arXiv:1809.01479*.

Stefan Heindorf, Martin Potthast, Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2017. Wsdm cup 2017: Vandalism detection and triple scoring. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 827–828. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining fact extraction and verification with neural semantic matching networks. *arXiv preprint arXiv:1811.07039*.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Dean Pomerleau and Delip Rao. 2017. Fake news challenge.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and verification (fever) shared task. *arXiv preprint arXiv:1811.10971*.

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.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 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.