IT University of Copenhagen
Natural Language Processing and Deep Learning

Stance Detection in News Articles Using Memory Networks

Amalie Arleth Viinholt, Terne Sasha Thorn Jakobsen, Nanna Bøgebjerg Hansen,
Adina-Ecaterina Negrău, Cristian-Cătălin Mitroi, Marija Stepanović

1 Introduction

This paper presents a re-implementation of an end-to-end memory network used for automatic stance detection from news data originally described in "Automatic Stance Detection Using End-to-End Memory Networks" by Mohtarami et al. [1]. It was developed as part of an ongoing effort to counteract the recent deluge of "fake news" on social media by exploiting advances in machine learning and natural language processing. Fake news is a type of false information created with an intention to deceive. It spreads through print, broadcast, and online news media, and unwittingly usually via social media. It is arguably one of the most pressing issues facing the news industry today as it not only undermines serious media coverage but also has the power to influence both personal decisions and major social and political events. Unfortunately, assessing the truthfulness of a news story is a painstaking and difficult task, even for trained experts. However, breaking it down into intermediate steps has shown promise for alleviating the task complexity. One such step is known as stance detection. Its goal is to identify the relative perspective of a piece of text toward a claim, which is why it is often considered a crucial step toward automated fact-checking [1, 2]. As a result, the Fake News Challenge, whose initial objective was to predict the veracity of a claim and provide evidence in support of the prediction (i.e. links to credible sources either confirming or disproving the claim), was revised and reduced to the more feasible task of stance detection [3].

Although they did not officially participate in the challenge, Mohtarami et al. propose a competitive solution to the stance detection task which integrates convolutional and recurrent neural networks, as well as a similarity matrix as part of a novel end-to-end memory network. Their model operates at the paragraph level and simultaneously predicts the stance of a document with respect to a given claim and extracts snippets of evidence for its predictions, which can be individual paragraphs or sequences of n-grams from a paragraph [1]. We have fully re-implemented their model according to the description provided in their paper and applied it to the same data set and stance detection task. We find that our re-implementation is unable to match the performance achieved by the original model. Nevertheless, our model outperforms the baselines in several important respects and is capable of extracting relatively meaningful snippets.

The purpose of this paper is, therefore, to re-implement the memory network, evaluate its performance on the stance detection task of the Fake News Challenge, and analyze the obtained results, by comparison with two baseline models and the original implementation. The first section will provide a brief summary of the original model and an overview of the Fake News Challenge, which will serve as a theoretical foundation of this paper. The subsequent sections will explore the research problem within the established theoretical framework, describe the methodology, namely, the corpus, feature extraction and representation, and model architecture, and present the obtained evaluation results. Finally, the last section will discuss the significance and shortcomings of our approach and highlight areas that require further improvement. [1]

[1] The full re-implementation can be found at: `https://github.com/AdinaNgr/NLPDL_Project_MemNN`

1.1 Notation and Abbreviations

MemNN	end-to-end memory network as defined in [4] and improved upon by [1]
$\mathbf{x}$	column vector $\mathbf{x} \in \mathbb{R}^d$
$\|\cdot\|$	Euclidean norm (square root of the sum of the squared components of a vector)
$(\cdot)^\mathsf{T}$	transpose (columns become rows and vice versa)
$\mathbf{a}^\mathsf{T}\mathbf{b}$	dot product (also scalar or inner product) of two vectors
$\mathbf{A}$	an n^{th}-rank tensor in m-dimensional space
$\mathbf{A} \odot \mathbf{B}$	Hadamard product (element-wise multiplication) of two tensors
$[\cdot, \cdot]$	concatenation of vectors
σ	the softmax function

2 Original Research and Related Work

2.1 Original Paper

The original paper chosen for this project is an attempt to improve on the stance detection task of the Fake News Challenge, which ran from December 2016 to June 2017. It uses an end-to-end memory network whose architecture integrates convolutional and recurrent neural networks with a similarity matrix for the purpose of both predicting the stance of a document with respect to a given claim and extracting snippets of text to serve as evidence for the prediction.

Their model is based on an end-to-end memory network developed by Sukhbaatar et al. [4], which consists of five essential components: (M, I, G, O, R), where the *memory* (M) contains object representations, the *input* (I) maps the input to its representation in M, the *generalization* (G) updates the memory with respect to new input, the *output* (O) produces an output for each new input and current memory state, and the final *response* component (R) converts the output into a desired response format, such as a textual response or class prediction. What gives the current model an advantage over the previous approaches to stance detection from news data is that, unlike standard memory networks, it features an additional component F, the *inference*. In this part the relevant pieces of the input are identified and passed forward to the generalization component where they are used to update the memory representations [1].

The input to the network is a document–statement pair which consists of a body text and a headline that may or may not belong to the same news article. The bodies are segmented into paragraphs each of which is compared to the claim made in the corresponding headline in order to determine the overall similarity between each body–claim pair and extract the most similar paragraph which should reveal evidence for the predicted stance of the body with respect to the claim. To achieve this, either part of the vectorized input pair is transformed separately into its memory representations using both a CNN and an LSTM. The rationale behind using both types of neural networks is that LSTMs are effective in capturing long-term dependencies throughout a lengthy input text, while CNNs emphasize the local context around words in the input sequence. For this reason, memory networks that rely on convolutional and recurrent neural networks are particularly well-suited for stance detection on textual news data where documents can be quite long and relevant information dispersed throughout the document [1].

After conducting an experimental evaluation of their model on the Fake News Challenge data set, the authors report state-of-the-art performance in comparison to several simpler models used as baselines. Their model evidently outperforms all the baseline models by achieving an accuracy score of 87.27, weighted accuracy of 78.97, and macro F_1-score of 56.75, which is on par with the system that won the Fake News Challenge. However, the most substantial contribution of their model, compared to the baselines and related work on the Fake News Challenge data, consists in its ability to explain its predictions. As previously mentioned, in addition to predicting the stance of the documents, this model also extracts short pieces of text from each article body which are treated as evidence for the predicted stance. This is a by-product of the inference component which computes cosine similarities between a claim and each paragraph from the corresponding article body at three different levels of representation. The extracted snippets represent the individual n-grams or multiple consecutive n-grams from an article that exhibit the highest similarity to the claim to which they are compared. Since the Fake News Challenge data set does not include annotations of this kind, the authors devised their own evaluation method for this task. Namely, they randomly sampled 100 agree and disagree document–claim pairs and qualitatively assessed the top five pieces of evidence that their model excerpted from each document. In 76 out 100 cases, their model correctly classified the examples and "provided arguably adequate snippets" [1]. In the long run, the authors hope their model will prove useful not only for stance detection and similar text classification tasks, but also for human experts who need to verify the factuality of a particular claim.

2.2 Related Work

The goal of the Fake News Challenge was to encourage researchers and practitioners to work on projects related to fact-checking news stories [2]. Fake news is an increasing problem, and there is much demand for automatic approaches to detecting it. However the task is not necessarily easy to solve [5]. As a result, the Challenge presents only a small part of the solution to fake news detection: a four-class stance detection task. Of the fifty participants who responded to the challenge, the top three teams, whose models achieved a weighted score of 82.02, 81.97, and 81.72 respectively, were named winner and runners-up [2]. However, a retrospective analysis of the challenge has brought the official evaluation metrics into question [6]. Since the classes in the data set are heavily skewed, the challenge introduced a point system that assigns 0.25 points for a correct binary classification of *unrelated–related*, and an additional 0.75 points for correctly predicting one of the related classes, *agree*, *disagree*, or *discuss*.[2] Although this seems like a good way to even out imbalanced data, the retrospective analysis shows that since the related classes are imbalanced as well, a model that predicts the majority related class, *discuss*, every time would achieve a higher score than the winning system of the challenge. For this reason, the authors propose a new evaluation metric: a class-wise, macro-averaged F_1-score. A system that perfectly classifies *unrelated* and always predicts *discuss* for related instances would reach an F_1-score of only 44.4 instead of a weighted score of 83.3.

In general, we notice the trend of combining different architectures inside one model to "take advantage of the complementarity of CNNs, LSTMs and DNNs" [7]. Nevertheless, the system that won third prize, and in retrospect scored better than the official winner with several hidden layers, used a multilayer perceptron with only one hidden layer [6]. This raises the possibility that a preferable solution might be to not only find a good model, but also the most informative input features. On the other hand, the model being reproduced in this paper takes a more complex approach to the task. As introduced in the previous section, their network is based on the end-to-end memory network by Sukhbaatar et al., which the authors describe as a "neural network with an explicit memory and a recurrent attention mechanism for reading the memory" [4]. In this work, they present both a single and multiple layer architecture, with multiple layers corresponding to having several stacked memory networks. Having shown promising results on two NLP tasks, language modeling and question answering, this type of architecture has also proven useful for determining the stance of long documents.

3 Implementation Methods and Results

In this section, we provide detailed descriptions of our corpus, input features, classification model and its performance results. We also make an effort to explain and justify our choice of methods regarding data processing and feature representation, as well as model architecture, optimization, and evaluation.

3.1 Corpus

The data set was published in 2017 as part of the first Fake News Challenge, whose aim was to explore how AI technologies can be used to oppose the influx of "fake news" [2]. Overall, the data is split across two different files. One of them contains article bodies and the other one headlines and stances corresponding to each body. As stated previously, a body-headline pair does not necessarily come from the same article. At the same time, a number of different headlines might be matched with the same body text and vice versa. Therefore, out of 49972 training instances, which is the total number of body–headline pairs in the training set, there are only 1683 unique body texts and 1648 unique headlines. Given a body–headline pair, the task is to learn a classifier that maps the input pair to one of four labels: *unrelated*, *discuss*, *agree*, or *disagree*, which represent the stance of the document with respect to the claim made in the headline. The *unrelated* class was generated by randomly matching headlines and documents belonging to different topics, which has had the unfortunate effect of inflating its share of the data set to 73%. The second largest class, *discuss*, which indicates that the article and the headline merely cover the same topic without the text taking a definite position on the claim, is also significantly larger than the remaining classes [6]. With almost 18% of training examples, the *discuss* class is two and a half times larger than the *agree* class with 7%, and eleven times greater than the *disagree* class, which gathers less than 2% of the training data. Thus, one of the most demanding aspects of this task is dealing with the highly imbalanced class distribution.

[2] Adopting terminology from the paper on the original memory network, we refer to this scoring system as "weighted accuracy" in Section 3.4 on Performance Evaluation.

3.2 Text Pre-Processing and Feature Representation

3.2.1 Pre-Processing

This section will describe the pre-processing procedures used to remove noisy elements from the text and, thus, facilitate both feature extraction and later interpretation of the model's performance. To begin with, each article body was split into paragraphs of which only the first nine paragraphs were preserved, while the rest were discarded. On the other hand, the articles that contained fewer than nine paragraphs were padded up to the same size. This had to be performed to allow all data to be represented in the form of a tensor, which requires all inputs to be of the same size. In accordance with the original implementation, the maximum number of paragraphs was set to nine because nine is the median number of paragraphs per article, and represents a compromise between too much padding and losing too much information by trimming.

From this point onward, both the articles and claims were processed in the same way. First, all characters were stripped of accents while other non-ASCII characters were removed altogether. The purpose of this was to both avoid Unicode decoding issues and remove special characters which were considered noise as most of the corpus was written in English. Additionally, we removed punctuation and replaced all URLs, hashtags, Twitter usernames, and numbers with their respective placeholder tokens, `<url>`, `<user>`, `<hashtag>`, and `<number>`, to make our input compatible with the vocabulary of the pre-trained GloVe embeddings [8]. Finally, we tokenized all paragraphs and claims and compiled our vocabulary using the tokens found in the training set as well as the additional `<unknown>` token to which we map all tokens that do not have a vector representation in the GloVe corpus as well as the tokens from the validation and test set that do not occur in the training set.

3.2.2 Feature Extraction and Representation

This section describes how the raw pre-processed textual data was converted into feature vectors that can be used by the network. Namely, all input document–statement (i.e. article body–claim) pairs are vectorized in two ways: one converts them into dense vector representations and the other into sparse bag-of-words representations. As previously explained, the dense word vectors were obtained from the GloVe corpus pre-trained on Twitter data. Following instructions from the original implementation, we map each word in the input sequence to its 100-dimensional GloVe vector. For this type of representation, however, it was necessary to specify the same value for the default paragraph and claim length to cope with the limitations of vector operations. Since this number was left unspecified in the description of the original implementation, we set it to 15 which seemed a reasonable trade-off between the median paragraph length of 27 words and median claim length of 10. Experimenting with larger lengths did not improve the model's performance. The outcome of this vectorization procedure were two tensors, one of rank 4, encoding the article bodies, and another of rank 3, encoding the claims. These two tensors were passed as input to the two LSTMs and CNNs, which the following section on model architecture will describe in more detail.

On the other hand, the sparse representations were obtained using a bag-of-words text vectorization method known as term frequency–inverse document frequency vectorization (tf–idf). This method counts the number of occurrences of each term in a corpus of documents and applies a tf–idf weighting scheme to the raw counts to give importance to terms whose frequency is high in a given document and low in the whole corpus, while scaling down the terms that appear in many documents. The result is a sparse matrix with one row per document and one column per unique term in the vocabulary. In our case, each paragraph and claim was treated as a separate document for tf–idf vectorization resulting in two sparse matrices, one containing vectorized paragraphs from every article body and the other containing vectorized claims. These two matrices were then used to compute the first similarity matrix which stores cosine similarities between a claim and each paragraph from the corresponding article body.

The advantage of sparse vectorization over dense word vectors comes from sparse vectorization not requiring paragraph or claim trimming, thereby taking into account all words from the input sequence. However, this is done at the expense of information regarding their order and potential syntactic and/or semantic relations. In this respect, dense representation is more useful as it allows us to consider n-grams in the input sequence without a significant increase in dimensionality, which will in turn make it possible to pinpoint the exact n-grams that contribute the most to each prediction. This is the reason dense vectors comprise the main input to the memory network, while sparse vectors serve merely as a supplement to the inference component.

3.3 Model Architecture and Optimization

Once the input has been prepared as described in the previous section, the tensors containing dense word representations of article bodies and claims can each be passed to the two neural network layers independently in order to obtain

their memory representations. Specifically, the bodies consisting of individual paragraphs are passed through a time-distributed LSTM layer with 100 units and *tanh* activation and a time-distributed CNN layer with 100 feature maps of width 5 and *ReLU* activation. The time-distributed functionality indicates that each paragraph in a body is treated as a different time-step to which the same transformation is applied. The obtained CNN memory representations are subsequently passed through a *maxout* layer, which takes the maximum across each of the feature maps computed by the CNN [9]. This allows us to keep track of 5-grams in each paragraph by representing each one compactly as a single number, which is simply the maximum of the 100-dimensional output of the convolutional layer. It should be noted that the LSTM and CNN memory representations are obtained and passed forward independently of each other allowing the model to "choose" which aspects of either representation are the most pertinent to the task at hand. As previously discussed, the model is expected to leverage the strong points of both types of representation: the potential of LSTMs to remember past information and the ability of CNNs to capture local relations. At the same time, the memory representations of the claims are obtained in the same way as those of the bodies, by passing them separately through an LSTM and CNN layer using the exact same parameters as detailed above but this time without time distribution as each claim is treated as a single textual statement without a temporal dimension. See Table 4 in Appendix A for a more detailed explanation of the model pipeline.

The resulting memory representations are passed to the inference component where the semantic similarity between a claim and each potential piece of evidence from the corresponding article is computed and used to update the existing memory representation. As mentioned before, we define semantic similarity between a claim and a paragraph as the cosine similarity between their vector representations, which is equal to their dot product normalized by the product of their Euclidean norms. In other words, if we denote a document array by $\mathbf{D}_i$, its constituent paragraph vectors by $\mathbf{x}_{ij}$, and the corresponding statement (claim) vector by $\mathbf{s}_i$, then we can quantify their similarity by measuring the cosine similarity of their vector representations:

$$\text{sim}(\mathbf{s}_i, \mathbf{x}_{ij}) = \frac{\mathbf{s}_i^\top \mathbf{x}_{ij}}{\|\mathbf{s}_i\| \, \|\mathbf{x}_{ij}\|}$$

We begin by computing the similarities between the sparse (tf–idf) representations of claims and article bodies. The obtained similarity matrix is then used to update the array containing the LSTM memory representation of article bodies by performing element-wise multiplication along the temporal axis (the dimension encoding individual paragraphs). This operation is followed by computing another similarity matrix, this time between the updated LSTM representation of bodies and the LSTM representation of claims, which is in turn used to update the array containing CNN memory representations of bodies. Finally, the third similarity matrix is calculated between the updated CNN representation of bodies and the CNN representation of claims. The alternation of these two operations constitutes the interchange between the inference and the generalization component. Please refer to Table 4, lines 29–34, for an exact and concise summary of these two components.

The updated memory representations and similarity matrices computed in the previous step are subsequently summarized and concatenated to produce the output memory representation (as shown in Table 4, line 37). To achieve this, we first summarize the CNN representation of article bodies by calculating the mean n-gram representation for each paragraph. In addition, we find the maximum and mean similarity between each paragraph and the claim for each of the three similarity matrices computed in the inference component. The maximum similarity identifies the paragraph that is most similar to the claim, whereas the mean similarity measures the overall similarity between the whole article and the claim. The resulting arrays are concatenated and passed forward to the response component together with the LSTM and CNN representations of claims.

As indicated above, the response component is produced by concatenating the output representations with the LSTM and CNN representations of claims (Table 4, line 40). The resulting tensor is then forwarded through a dense layer with 300 hidden units and *ReLU* activation. The network is afterward regularized by means of *dropout*, which curbs overfitting by randomly deactivating 70% of input units. Finally, another fully connected layer, this time with *softmax* activation, is applied to the result of the previous layer to obtain the stance predictions. The whole procedure is distilled in lines 41–43 of Table 4. Note that the original implementation did not include the 300-unit dense layer and subsequent dropout, but, in our experience, that model was more susceptible to overfitting.

In the end, the model is trained for 15 epochs in batches of 128 instances and validated on 20% of the held-out training data. We tried several simple random-sampling strategies to mitigate class imbalance, such as undersampling the majority classes, oversampling the minority classes, and combining both under- and oversampling. The only strategy that yielded performance results that were on a par with the baselines was oversampling in which all but the majority class (i.e. the three related classes) were oversampled randomly with replacement until their size equaled the size of the majority class. The original memory network was trained using an undersampling technique which randomly undersamples all but the minority class at each training iteration. While we agree that removing the excess instances from the majority

classes should be an optimal way of improving the visibility of the minority class, in this case, any form of undersampling has had an adverse effect on our model's performance, especially the ones that purge over 50% of the training data.

In the last step of our architecture, we extract meaningful snippets. These are used to provide evidence for a given predicted stance. It is this feature that makes this approach special – none of the other models provide proof for a given document–claim combination. In this step we rely on the CNN's capacity to detect local patterns. More precisely, we first obtain the most similar paragraph from the $\mathbf{P}_{\mathbf{cnn}}$ component (line 34 in appendix A). We then extract the most relevant *n*-gram from the CNN's *maxout* layer $\mathbf{D}_{\mathbf{cnn}}$ (line 23 in appendix A). Examples of these are shown in Table 3.

3.4 Performance Evaluation

In this section, we present the evaluation metrics used to assess the performance of our model in comparison to the available baselines. Namely, we use three different evaluation methods: accuracy, F_1, and weighted accuracy, which was the official scoring metric used in the Fake News Challenge. As described previously, this metric is a two-level scoring system in which classifying a body–claim pair correctly as either *unrelated* or *related* counts as 0.25 points and classifying a *related* pair correctly, as either *discuss*, *agree*, or *disagree*, counts as an additional 0.75. The weighted accuracy is then obtained by dividing the raw scores by the maximum possible score on the test set [2].

Due to the fact that the data set is very imbalanced it is difficult to get a trustworthy impression of how well the model performs. Therefore, it is recommended to compare it to baseline models. We have chosen two baselines: the simplest one that always predicts the majority class (*unrelated*), and Gradient Boosting, which is the baseline used in the Fake News Challenge [2]. The results of our model, the original memory network, and the baselines can be seen in Table 1.

Models	Weighted Accuracy	F_1	Accuracy
All-unrelated	39.37	20.96	72.20
Gradient Boosting	75.20	46.13	86.32
Our MemNN	**74.85**	**50.20**	**83.93**
Original MemNN	78.97	56.75	87.27

Table 1: Evaluation results compared with the baselines and original memory network.

In addition to the above metrics, we also present our model's precision, recall, and F_1 scores (Table 2) and a normalized confusion matrix (Figure 1), which provide a deeper understanding of our model's performance.

Label	Precision	Recall	F_1
unrelated	94.13	95.58	94.85
discuss	61.71	71.48	66.24
agree	43.13	30.53	35.75
disagree	7.2	2.73	3.95
Macro average	51.54	50.08	50.20

Table 2: Precision, recall, and F_1 scores for each class and their average.

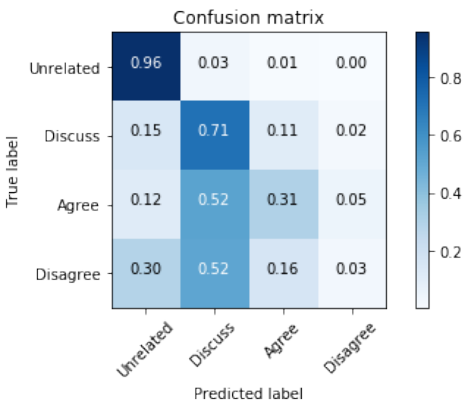

Figure 1: Normalized confusion matrix.

When it comes to the output of the inference component, Table 3 shows a random selection of claim–evidence pairs which were classified correctly by our network.[3] We only examine the pairs belonging to the *agree* or *disagree* class, as they are deemed more relevant for the stance detection task. Each pair consists of a claim and the most similar paragraph from the corresponding article body, which is considered evidence for the predicted stance. As explained before, since evidence extraction was not part of the Challenge, we have no way of quantifying the performance of our model with respect to this task. Therefore, we only provide a qualitative analysis of the results (see Discussion).

Stance	Claim	Evidence snippet
agree	rise of the <unknown> cat pet who was killed and buried by his owner climbed	doctors at the humane society of tampa bay are scheduling surgery to save <unknown> life
agree	<unknown> cat is getting gender assignment surgery	the cat has both male and female genitalia
agree	wife cuts off husband 's penis twice after catching him cheating	grabbing a pair of scissors she stormed into their bedroom where he was sleeping and
agree	florida woman gets plastic surgery to add a third <unknown>	i got it because i wanted to make myself <unknown> to men because i do
agree	no michael jackson isnt bruno mars dad	however the claims in the story are untrue no such private information was leaked and
agree	climate change the hoax that costs us <number> billion a day	last year climate change business journal calculated that the total annual spend on the climate
disagree	<unknown> eyes ban saudi women with pretty eyes made to cover them up	a new law in saudi arabia forcing women to wear full <unknown> if their eyes
disagree	bear attack foiled by justin biebers music a story too good to check	justin biebers song has saved a russian man from being <unknown> by a bear

Table 3: Examples of *agree* and *disagree* claim–evidence pairs which were classified correctly by our model and extracted automatically by the inference component. The evidence snippet represents the most similar paragraph from the article that corresponds to the given claim, while the highlighted text signifies the most similar 5-gram as extracted by the CNN.

4 Discussion

This section will discuss the strengths and shortcomings of the presented stance classification model, as well as offer suggestions for possible improvement. We begin with quantitative evaluation by examining the classification results presented in the previous section. From Table 2, we observe that the model is the most accurate at predicting the *unrelated* class, classifying correctly 95.58% of the total unrelated body–claim pairs. The second most accurately predicted class is *discuss* with 71.48% true positives. However, the lower precision scores for these two classes indicate that they are being over-predicted at the expense of the remaining two classes. Indeed, this is apparent from the precision and recall scores for *agree* and *disagree*, as well as the overall confusion matrix. Namely, the model is quite ineffective at predicting these two classes, especially the *disagree* class, to which it seems almost completely insensitive, and frequently confuses them with the *discuss* class. This is to be expected mostly as a consequence of severe class imbalance and was experienced to some extent by all the models that had previously undertaken this task. Unfortunately, in stance detection, the two minority classes are the most relevant, which means that their recall should be prioritized over precision. One way to achieve this is by undersampling the majority class(es). However, this sampling strategy proved detrimental to our model's overall performance as undersampling had severely reduced the size of the training set. This underscores the need for experimentation with more advanced undersampling methods, such as Cluster Centroids, Edited Nearest Neighbors, and Tomek Links, as well as development of more adequate solutions to learning from imbalanced data.

Compared with the baseline estimators, our memory network clearly outperforms the majority baseline. The overall accuracy of the Gradient Boosting model is higher than our model's, but the higher F_1 score of our memory network indicates that it is more accurate at predicting the minority classes. However, for reasons we are unable to ascertain, our model does not measure up to the original memory network by Mohtarami et al. As their paper offers little information on feature representation, one possible explanation is that the discrepancies in performance stem from the differences in text processing and vectorization. Another explanation is that the discrepancies originate from the differences in sampling strategies used to overcome class imbalance. The most plausible explanation, though, is that, despite our best effort, we were unable to perfectly recreate their model solely based on verbal description, as some assumptions the

[3] The complete table can be found inside the file `examples.csv` at `https://github.com/AdinaNgr/NLPDL_Project_MemNN`

authors seem to make in their implementation were not made explicit in their paper.

We proceed to the qualitative analysis of the claim–evidence pairs produced by the inference component. Based on our observations, most of the extracted snippets exhibit high semantic similarity to their claims. While this may be enough to discriminate between related and unrelated pairs, it is unlikely that word similarity alone (measured as cosine similarity) can be used to distinguish agreement or disagreement from mere relatedness. On the contrary, discerning stance from text requires a more profound understanding of language, one that includes handling negation, ambiguity, iconicity, metaphor, domain-specific jargon, non-standard usage and slang, as well as extra-linguistic context. Many of these concepts can pose problems even for trained linguists. In fact, as the retrospective analysis has shown, when asked to manually label 200 instances, human raters reach a relatively low inter-annotator agreement on the three related classes ($\kappa = 0.218$), indicating that the task is difficult even for humans [6]. In addition to class imbalance, this can also be used to explain the model's inability to recognize these two stances, as well as why it frequently mistakes them for *discuss*. One example where cosine similarity is particularly unhelpful is when used to determine the stance of a pair in which both the claim and the article consist largely of placeholder tokens, such as `<unknown>` or `<number>`. In such cases, the inference component might report high similarity even between unrelated pairs. In light of these findings, we recommend a more studious approach to text processing and feature engineering, relying on domain knowledge and manual inspection of the training data. For example, using a list of hand-crafted cue words that are indicative of the author's stance and certainty in journalism, and categorized according to semantic content (e.g. *knowledge*, *belief*, *report*, *doubt*, *denial*, etc.), has shown promising results in stance detection from Twitter data and news [10], [11].

5 Conclusion

In this paper we have performed stance detection on the Fake News Challenge data set using a novel end-to-end memory network that jointly infers the stance of a news article with respect to the claim in a corresponding headline and highlights relevant snippets of evidence in support of its predictions. We combine multiple techniques and model architectures into one pipeline – LSTMs, CNNs, DNNs, pre-trained word vectors, bag-of-words vectorization based on tf–idf, and cosine similarity – taking advantage of each of their strengths. Although we were unable to reproduce the results achieved by the original implementation, our F_1 scores demonstrate a significant improvement over the baselines. Furthermore, the evidence snippets based on word similarity provide concrete explanations of the model's predictions, a feature unique to this memory network. Nonetheless, future stance detectors would undoubtedly benefit from larger news corpora, exhaustive feature selection and careful engineering, and state-of-the-art findings in the areas of NLP, deep learning, and machine learning from imbalanced data.

A Memory Network Architecture

Input:
a document (article body) $\mathbf{D}$ segmented into paragraphs (potential pieces of evidence) $\mathbf{X_j}$
a textual statement $\mathbf{S}$ containing a claim (article headline)

Output:
the stance of a document with respect to the corresponding claim (*agree*, *disagree*, *discuss*, *unrelated*) $\hat{\mathbf{y}}$

Inference Output:
most similar paragraphs with their similarity scores

1. Input Encoding / Vectorization

Dense Representation: word embeddings pre-trained on Twitter data (GloVe)
$\mathbf{D} = (x, w, e)$
$\mathbf{S} = (w, e)$

Sparse Representation: term frequency–inverse document frequency
$\mathbf{D_{tfidf}} = (x, v)$
$\mathbf{s_{tfidf}} = (v)$

2. Memory Representation

$$\mathbf{D} \xrightarrow{\text{time-distributed LSTM (100 units)}} \mathbf{D_{lstm}}$$
$$\mathbf{D} \xrightarrow{\text{time-distributed CNN (100 filters, size 5)}} \mathbf{D_{cnn}}$$
$$\mathbf{D_{cnn}} \xrightarrow{\text{MaxOut}} \mathbf{D_{cnn}}$$

$$\mathbf{S} \xrightarrow{\text{LSTM (100 units)}} \mathbf{s_{lstm}}$$
$$\mathbf{S} \xrightarrow{\text{CNN (100 filters, size 5)}} \mathbf{S_{cnn}}$$
$$\mathbf{S_{cnn}} \xrightarrow{\text{MaxOut}} \mathbf{s_{cnn}}$$

3. Inference and Generalization

$$\mathbf{p}^{j}_{tfidf} = \frac{\mathbf{s_{tfidf}}^{\intercal}\, \mathbf{x}^{j}_{tfidf}}{\|\mathbf{s_{tfidf}}\|\, \|\mathbf{x}^{j}_{tfidf}\|}$$
$$\mathbf{D_{lstm}} = \mathbf{D_{lstm}} \odot \mathbf{p_{tfidf}}$$
$$\mathbf{p}^{j}_{lstm} = \frac{\mathbf{s_{lstm}}^{\intercal}\, \mathbf{x}^{j}_{lstm}}{\|\mathbf{s_{lstm}}\|\, \|\mathbf{x}^{j}_{lstm}\|}$$
$$\mathbf{D_{cnn}} = \mathbf{D_{cnn}} \odot \mathbf{p_{lstm}}$$
$$\mathbf{p}^{j}_{cnn} = \frac{\mathbf{s_{cnn}}^{\intercal}\, \mathbf{x}^{j}_{cnn}}{\|\mathbf{s_{cnn}}\|\, \|\mathbf{x}^{j}_{cnn}\|}$$

4. Output Memory Representation

$$\mathbf{o} = [\text{mean}(\mathbf{D_{cnn}}),\ \max(\mathbf{p_{cnn}}),\ \text{mean}(\mathbf{p_{cnn}}),\ \max(\mathbf{p_{lstm}}),\ \text{mean}(\mathbf{p_{lstm}}),\ \max(\mathbf{p_{tfidf}}),\ \text{mean}(\mathbf{p_{tfidf}})]$$

5. Final Response (Class Prediction)

$$\mathbf{r} = [\mathbf{o},\ \mathbf{s_{lstm}},\ \mathbf{s_{cnn}}]$$
$$\mathbf{r} \xrightarrow{\text{MLP (300 units, ReLU)}} \mathbf{r_{mlp}}$$
$$\mathbf{r_{mlp}} \xrightarrow{\text{DropOut (0.7)}} \mathbf{r_{mlp}}$$
$$\mathbf{r_{mlp}} \xrightarrow{\text{MLP (4 units, } \sigma)} \hat{\mathbf{y}}$$

Table 4: Architecture of the re-implemented memory network

References

[1] M. Mohtarami, R. Baly, J. R. Glass, P. Nakov, L. Màrquez, and A. Moschitti, "Automatic stance detection using end-to-end memory networks," in *NAACL-HLT*. Association for Computational Linguistics, 2018, pp. 767–776. [Online]. Available: https://arxiv.org/abs/1804.07581

[2] D. Pomerleau and D. Rao, "The fake news challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news," http://www.fakenewschallenge.org/, 2017, accessed: 2010-09-30.

[3] Z. C. Lipton, "Fake news challenge – revised and revisited," *Approximately Correct: Technical and Social Perspectives on Machine Learning*, Feb 2017. [Online]. Available: http://approximatelycorrect.com/2017/02/26/fake-news-challenge-revised-and-revisited/

[4] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448. [Online]. Available: https://arxiv.org/abs/1503.08895

[5] G. Simons, "Fake news: As the problem or a symptom of a deeper problem?" *Obraz/Image*, 2018.

[6] A. Hanselowski, A. PVS, B. Schiller, F. Caspelherr, D. Chaudhuri, C. M. Meyer, and I. Gurevych, "A retrospective analysis of the fake news challenge stance-detection task," in *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 1859–1874. [Online]. Available: http://aclweb.org/anthology/C18-1158

[7] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4580–4584, 2015.

[8] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[9] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of ICML*, 2013, p. 1319–1327.

[10] H. Bahuleyan and O. Vechtomova, "Uwaterloo at semeval-2017 task 8: Detecting stance towards rumours with topic independent features," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, August 2017, pp. 461–464. [Online]. Available: http://www.aclweb.org/anthology/S17-2080

[11] B. Ghanem, P. Rosso, and F. Rangel, "Stance detection in fake news a combined feature representation," in *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, 2018, pp. 66–71. [Online]. Available: http://aclweb.org/anthology/W18-5510