Fake News Disambiguation: Why Is It Fake?

Karish Grover
Roll No. 2019471
IIIT Delhi
karish19471@iiitd.ac.in

Nirali Arora
Roll No. Phd21002
IIIT Delhi
niralis@iiitd.ac.in

Aisha Aijaz
Roll No. PhD21011
IIIT Delhi
aishaa@iiitd.ac.in

Abstract

This project considers the fake news detection problem under a more realistic scenario on social media. Given just a source short-text tweet, we aim to predict whether the given tweet is fake or not. We further aim to generate an explanation for this prediction. This paper presents **Attention-enhanced Multi-channel Recurrent Convolutional Network** (**AMRCN**), for explainable fake news detection. We explain our final predictions by highlighting the essential words in the short tweet text. Experimental results and extensive ablation studies show that our model outperforms the baseline systems on two benchmarking datasets.

1 Introduction

Social media has become a necessitous part of people's lives. Users can interact, express their views, comment on others' views, and access news through various social media platforms. Some people make unjust use of these mediums to spread misinformation. As fake news pieces are intentionally created to spread inaccurate information and lure people into believing them, they often have opinionated and sensational language styles, which can help verify the integrity of the news piece. In addition, a news document contains linguistic cues at different levels, such as word-level and sentence-level, which provide different degrees of importance for the explainability of why the news is fake. For example, consider an unreliable news piece, Iranian woman jailed for fictional unpublished story about a woman stoned to death for adultery. The words jailed and fictional contributes more in deciding whether the news claim is fake or not, as compared to the other words in the sentence. Further, as shown in Figure 1, the highlighted words are more crucial towards the final prediction as compared to the others parts of the news piece as

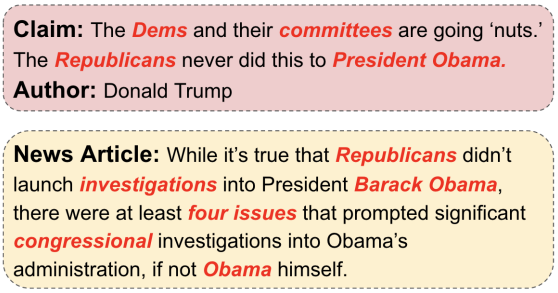

Figure 1: Motivating example behind highlighting the relative importances of words towards final prediction as fake/real.

they give more information about the writing style of the news piece.

2 Related Work

There are broadly three existing approaches for detecting/curbing fake news, (1) News Content-based, (2) Propagation-based, and (3) Social/User context- based. Typical methods rely on the content of news articles/tweets to extract the textual features, such as TFIDF/n-gram, and apply supervised learning (e.g., random forest, support vector machine, and Naive Bayes) for binary classification (Shu et al., 2017). State-of-the-art approaches like (Sheng et al., 2021) jointly model the news content to extract user- and fact- preference maps. Some consider the polarization in the social network between trustworthy and non-trustworthy users and observe the patterns in news diffusion (Monti et al., 2019). AENet (Jain et al., 2021) uses attention enabled neural architecture using contextual features related to the user (e.g., location, time of the speech, etc.) and the news piece to detect fake news. There are still many challenges in detecting fake news using these methods. Table 4 lays out such challenges.

Attention-based mechanisms have proven to be

Table 1: Statistics of two Twitter datasets.

	Twitter15	Twitter16
# source tweets	741	411
# true	372	206
# fake	369	205
min. #words/tweet	1	3
avg. #words/tweet	11.268	11.002
max. #words/tweet	37	22

very effective because of their ability to stress upon the most important parts of a text. (Roy et al., 2018) used deep learning ensemble architecture based on CNN and Bi-LSTM to capture hidden features and information in both directions, respectively, and (Long, 2017) proposed a hybrid attention-based LSTM model. In (Popat et al., 2018), the authors concatenate the claim text and content of the article and apply word-level self-attention to distinguish between real and false claims.

Almost none of the existing methods don't focus on the explainability of their prediction. Although dEFEND (Shu et al., 2019) can generate a decent explanation by following a co-attentive mechanism and highlighting explanatory user comments, it requires a long text of source articles and a set of user comments, which are difficult to obtain. Hence, we take up a realistic scenario where data would be readily available to overcome these challenges. We aim to explain the final prediction.

3 Problem Statement

Given the short text tweet content, we aim to classify this tweet as fake or real, i.e., binary classification. Further, we aim to explain why this tweet is fake/actual by highlighting tokens/phrases from the tweet content that contribute relatively more to the final prediction.

4 Dataset Collection and Preprocessing

We utilize two well-known datasets compiled by (Ma et al., 2017), Twitter15 and Twitter16, for our work. These datasets contain source tweets and their corresponding sequences of retweet users. The dataset statistics have been laid out in Table 1. These datasets are balanced towards the classes and consist of short length tweets. We remove the stop words from the tweets, replace all URLs with the token `URL`, and stem the tweets in the dataset for proceeding with the experiments.

5 Baseline Models

We evaluate the dataset collected on several baselines., as listed in Table 2. (1) **NB** (Multinomial Naive Bayes), (2) **LR** (Logistic Regression), (3) **DT** (Decision Trees), (4) **SVM** (Support Vector Machines), (5) **RF** (Random Forest - Bagging), (6) **XGB** (XGBoost - Boosting), (7) **CNN**(Convolutional Neural Networks), (8) **LSTM** (Long Short Term Memory cells), (9) **GRU** (Gated Recurrent Units), (10) **Bi-RNN** (Bi-directional Recurrent Neural Networks), (10) **RCNN** (Recurrent Convolutional Neural Networks). The neural network baselines have an embedding layer containing the 100D Glove embeddings of the tokens. We use F1 score, accuracy, precision and recall metrics to get a better idea of the models' performance on both the datasets. All the models were trained on a 80:10:10 train-test-val split of the dataset.

5.1 Ablation and Analysis

For the simple machine learning models (1-5), we use TFIDF vectors to transform the input tweet. We perform experiments with 4 types of features in this case **(a)** Count Vectors, **(b)** Word-level TFIDF, **(c)** Character-level TFIDF, and **(d)** N-Gram-level TFIDF vectors. We observe that in the simple ML models, the Count vectors and word-level TFIDF vectors lead to better F1 scores than the other two. Further, we can see that NB, SVM and LR perform better on both datasets than the Bagging and Boosting models. This is probably because of the simplicity of the former models in handling short length tweets. In the case of CNN, we experiment by using both average and max pooling, adding a dropout layer, and making the embedding layer trainable. It was observed from the results that max-pooling gives better performance than average pooling in terms of the F1 score.

Making the embedding layer *trainable* boosts performance in the case of LSTM, GRU, BiGRU and RCNN. This is intuitive because the initial Glove embeddings get fine-tuned on the dataset in hand when making the embedding layer trainable. However, if we consider a general scenario, then making the embeddings trainable would not be perfect because it would make the model highly dataset-specific.

Further, in RCNN, using multiple CNNs instead of 1 increases the performance because a hierarchical CNN structure has added benefit of capturing sequential correlation and more linguistic cues. Us-

Table 2: Baseline results and ablation study.

	Twitter15				Twitter16			
Method	**Acc**	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
NB *w/* Countvectors	0.9301	0.9404	0.9080	0.9239	0.8932	0.8444	0.9047	0.8735
NB *w/* Word TFIDF	0.9247	0.9523	0.8888	0.9195	0.9029	0.8888	0.8888	0.8888
NB *w/* NGram Vectors	0.9086	0.9523	0.8602	0.9039	0.8640	0.9333	0.7924	0.8571
NB *w/* CharVectors	0.8924	0.9523	0.8333	0.8888	0.9126	0.9555	0.8601	0.9052
LR *w/* Countvectors	0.9086	0.9047	0.8941	0.8994	0.8932	0.8222	0.9251	0.8705
LR *w/* Word TFIDF	0.9301	0.9047	0.9382	0.9212	0.9029	0.8444	0.9268	0.8837
LR *w/* NGram Vectors	0.9408	0.9047	0.9620	0.9325	0.8252	0.8888	0.7547	0.8163
LR *w/* CharVectors	0.9086	0.8928	0.9036	0.8982	0.9126	0.8222	0.9736	0.8915
SVM *w/* Countvectors	0.9354	0.9047	0.9501	0.9268	0.9126	0.8444	0.9501	0.8941
SVM *w/* Word TFIDF	0.9408	0.9047	0.9620	0.9325	0.8155	0.9111	0.7321	0.8118
SVM *w/* NGram Vectors	0.9193	0.8452	0.9726	0.9044	0.8252	0.9555	0.7288	0.8269
SVM *w/* CharVectors	0.9408	0.9166	0.9506	0.9333	0.9223	0.8444	0.9743	0.9047
RF *w/* Countvectors	0.9354	0.8809	0.9736	0.9250	0.9029	0.8001	0.9729	0.8780
RF *w/* Word TFIDF	0.9193	0.8452	0.9726	0.9044	0.8834	0.7777	0.9459	0.8536
RF *w/* CharVectors	0.8978	0.8809	0.8915	0.8862	0.9223	0.8444	0.9743	0.9047
XGB *w/* Countvectors	0.8655	0.7738	0.9154	0.8387	0.8155	0.7555	0.8095	0.7816
XGB *w/* Word TFIDF	0.8602	0.7619	0.9142	0.8311	0.8446	0.7555	0.8717	0.8095
XGB *w/* CharVectors	0.8548	0.8690	0.8202	0.8439	0.9029	0.8666	0.9069	0.8863
CNN	0.9247	0.9074	0.9607	0.9301	0.8155	0.8001	0.7826	0.7912
CNN *w/o* Dropout	0.9139	0.8796	0.9693	0.9223	0.8446	0.8222	0.8222	0.8222
CNN *w/* TrainEmb	0.9193	0.8796	0.9793	0.9268	0.8252	0.8222	0.7872	0.8043
CNN *w/* AvgPool	0.8978	0.8703	0.9494	0.9082	0.7864	0.7111	0.7804	0.7441
LSTM	0.8602	0.8611	0.8942	0.8773	0.7669	0.8001	0.7058	0.7501
LSTM *w/* TrainEmb	0.8978	0.9166	0.9082	0.9124	0.8834	0.8222	0.9024	0.8604
GRU	0.8333	0.8333	0.8737	0.8530	0.8737	0.7777	0.9210	0.8433
GRU *w/* TrainEmb	0.9139	0.9074	0.9423	0.9245	0.8737	0.8666	0.8478	0.8571
BiGRU	0.8602	0.8055	0.9456	0.8701	0.8349	0.7111	0.8888	0.7901
BiGRU *w/* TrainEmb	0.9032	0.8981	0.9326	0.9150	0.8932	0.8444	0.9047	0.8735
RCNN *w/* TrainEmb	0.8590	0.8571	0.8450	0.8510	0.8657	0.8101	0.9275	0.8648
RCNN *w/* Multiple CNN	0.9127	0.9189	0.9066	0.9127	0.9261	0.8676	0.9672	0.9147

ing the dropout layer with CNN reduces the F1 score, leading to information loss in short-length tweets.

5.2 Giving Explanation in Baselines (Naive)

While we discuss fake news explainability using the proposed model architecture in the subsequent sections, in this section, we use ELI5[1] to demystify the predictions made by our baseline models. ELI5 is a Python package that helps to debug machine learning classifiers and explain their predictions. We use the TextExplainer[2] algorithm to explain predictions of the baseline classifiers using LIME algorithm (Ribeiro et al., 2016).

[1] https://bit.ly/eli5_
[2] https://bit.ly/TextExplainer

Figure 2 shows 5 examples for the trained SVM classifier. Features/words with positive correlations with the target are shown in green, otherwise red. This gives us an idea about the most important words contributing to the final prediction. More information about the LIME algorithm has been mentioned in the Appendix A, since it's beyond the main focus of this project.

6 Methodology

The model architecture is shown in Figure 3. It can be divided into four broad parts: (1) CNN-based Representation, (2) Attention enhanced Word-level Encoder, (3) Multiple Channels, and (4) Explainability of the Prediction.

Figure 2: Highlighting using Eli5.

6.1 CNN-based Representation

Convolutional Neural Networks (CNN) are good at learning the sequential correlation. We use this to our advantage. The input glove embeddings of the tokens are passed through a 1D Convolutional layer. We consider λ consecutive tokens at one time to model their sequential correlation (Lu and Li, 2020), i.e., $\langle \mathbf{x}_t, ..., \mathbf{x}_{t+\lambda-1} \rangle$. Hence the filter is set as $\mathbf{W}_f \in R^{\lambda \times v}$. Then the output representation vector $\mathbf{C} \in R^{d \times (t+\lambda-1)}$ is given by

$$\mathbf{C} = ReLU(\mathbf{W}_f \cdot \mathbf{X}_{t:t+\lambda-1} + b_f) \qquad (1)$$

where $\mathbf{W}_f$ is the matrix of learnable parameters, $ReLU$ is the activation function, $\mathbf{X}_{t:t+\lambda-1}$ depicts sub-matrices whose first row's index is from $t = 1$ to $t = n - \lambda + 1$, and b_f is the bias term.

6.2 Word Encoder

We learn the sentence representation via a recurrent neural network (RNN) based word encoder. Though theoretically, RNN is able to capture long-term dependencies, but in practice, it leads to information bottleneck and vanishing gradients and it seems to "forget" the information. To capture long-term dependencies of RNN, we employ a bidirectional GRU. GRUs are known for a more persistent memory.

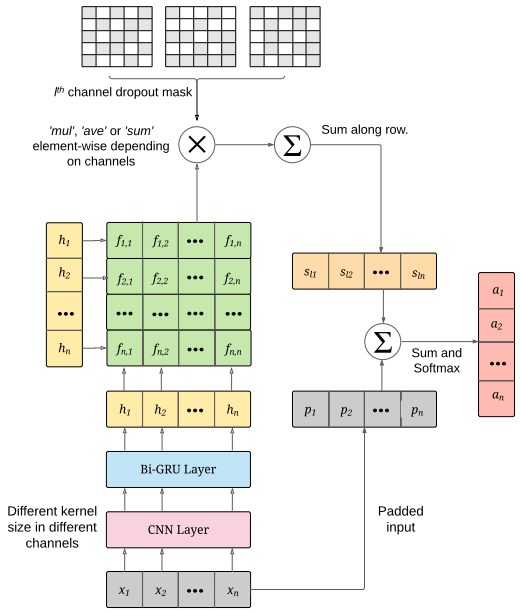

Figure 3: Model Architecture.

The CNN-based representation is fed into the word encoder. The word-level representations are obtained by concatenating the forward and backward hidden states of the BiGRU. This helps in capturing the word-level context. Next, by using a one-layer MLP and softmax function, we calculate the normalized importance (attention) weights over the word representations generated by the word encoder. The Figure 3 shows the attention mechanism along with the model architecture used.

6.3 Multiple Channels

We have three different channels in the final model which achieves the highest performance. These three channels start with CNN based representations with kernel sizes 3, 4 and 5 respectively, followed by Bi-Directional GRU with merge modes as `concat`, `sum`, and `ave` respectively. Further, they have dropout layers with different dropout ratios, i.e. 0.5, 0.4, and 0.2, respectively. Finally, all these three channel outputs are concatenated together.

6.4 Explainability

We use the attention weights at the end of the model pipeline to highlight the most relevant words for detecting whether a piece of news is fake or not. A visualization of these attention weights is shown in Figure 5 and Figure 4 for 4 example input tweets from the Twitter15 dataset. The Figure 4 shows the attention weights in a model without channels, and

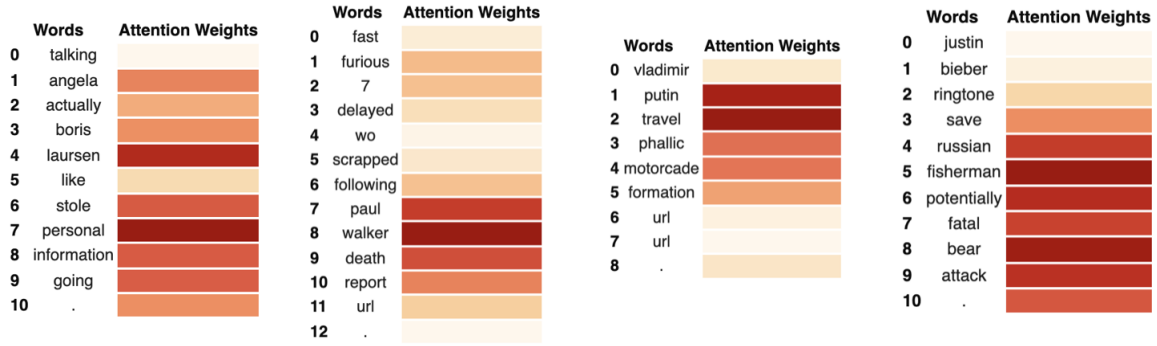

Figure 4: Highlighting using No-channels.

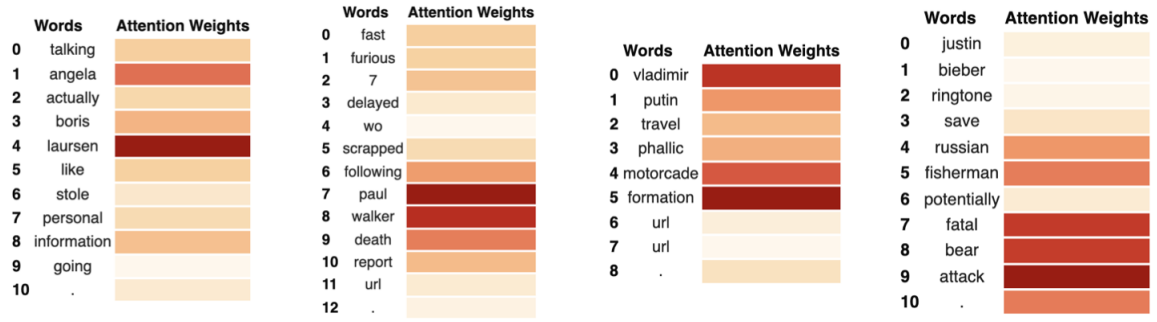

Figure 5: Highlighting using Channels.

Table 3: Our model results. The best model results are highlighted by **bold**.

	Twitter15				Twitter16			
Method	**Acc**	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
AMRCN *w/* 1 CNN	0.8783	0.9189	0.8501	0.8831	0.8751	0.8501	0.8947	0.8717
AMRCN *w/* 2-HCNN	0.9054	0.9459	0.8751	0.9090	0.7751	0.7501	0.7894	0.7692
AMRCN *w/* 3-HCNN	0.8918	0.9459	0.8536	0.8974	0.8501	0.9001	0.8181	0.8571
AMRCN *w/* 'ave' merge	0.8918	0.8648	0.9142	0.8888	0.9251	0.9001	0.9473	0.9230
AMRCN *w/* 'sum' merge	0.8783	0.9189	0.8501	0.8831	0.9002	0.8501	0.9444	0.8947
AMRCN *w/o* CNN	0.8918	0.9189	0.8717	0.8947	0.8783	0.8918	0.8684	0.8803
AMRCN *w/o* BiDirectional	0.8783	0.8378	0.9117	0.8732	0.8513	0.8648	0.8421	0.8533
AMRCN *w/* Chan *w/o* CNN	0.9054	0.8648	0.9411	0.9014	0.9189	0.8648	0.9696	0.9142
AMRCN *w/* Channels	**0.9324**	**0.9459**	**0.9210**	**0.9333**	**0.9459**	**0.9189**	**0.9714**	**0.9444**

Figure 5 shows the distribution of weights in the final model with 3 channels. As can be seen from the figures, the weights in the case of the model with channels are more concentrated, which gives a more precise explanation. However, when we remove channels, the weights are more spread across the tweet text, which is less precise in highlighting the most important words. This gives a visual justification for the utility of multiple channels in the model architecture. The attention weights as seen in the visualization are shown after taking the natural logarithm of the weights. This is because some the weights are larger compared to others, and this leads to difficulty in visualization. Taking the `log`, helps in normalization.

As it can be seen from the example visualization above, in the sentence "fast and furious 7 scrapped following paul walker's death report", the words "paul", "walker", and "death" are the most significant towards the final prediction that it is a True piece of information. Similarly in the other example, words like "fatal", "bear", and "attack" are more relevant. This use of attention weights to give explanations is quite reasonable.

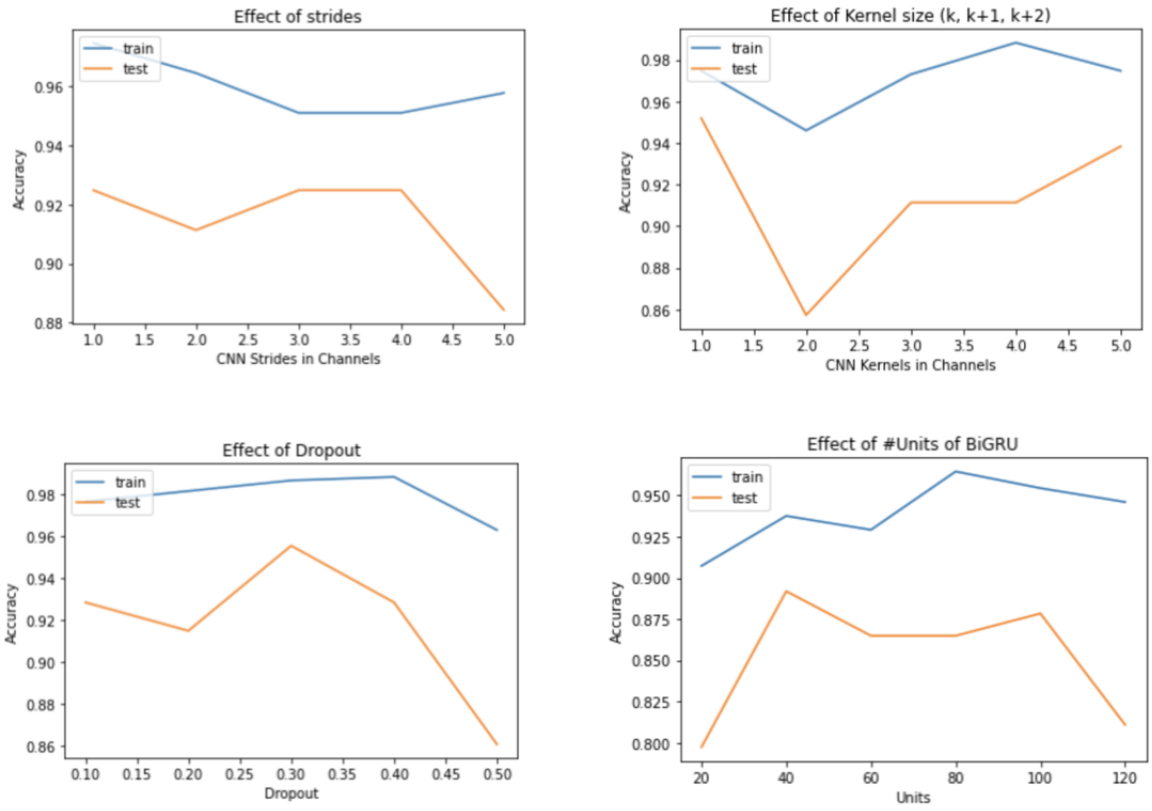

Figure 6: Effect of Hyperparameters.

7 Model Results and Analysis

Table 3 lays down the ablation study for our model AMRCN. The final model AMRCN *w/* Channels outperforms all the baseline models in terms of the prediction F1 score. In addition to generating explainability as discussed in Section 6.4, AMRCN performs better than the baseline models in the binary classification task. This highlights that attention (i.e. explainability) has a side-product of boosted performance, and is intuitive. If a model can explain its prediction, it will directly or indirectly make better predictions. Our attention-enhanced framework proves to be quite effective in giving explanations and, at the same time distinguishing between fake and real tweets.

In the ablation study, as shown in Table 3, it is observed that if we remove CNN from the model, it degrades the performance. This supports our hypothesis that CNN-based representation helps capture the sequential correlation and thus boosts performance. Further, we can also see the importance and effect of using multiple channels on the model performance, as AMRCN without channels performs poorly compared to the final model. All the variants of AMRCN introduced in the ablation table perform comparably with the baseline models.

Further, we can see that the performance boosts on replacing a simple CNN with a Hierarchical CNN. This is primarily because HCNN can capture more linguistic cues. Also, in the case of the merge mode in the bi-directional GRU (how to merge the hidden state representations from forwards and backward layers), we tried summing, averaging and simple concatenation. The averaging merge mode performs better than the other two. These three modes are used in parallel in the case of our final model with three channels. We also observed that on replacing the BiGRU with a uni-directional GRU, the F1 score drops because BiGRU helps capture both forward and backward context and thus would naturally give a better performance.

7.1 Effect of Hyper-parameters

We observe the impact of hyperparameters on our final model on the Twitter15 dataset. As we vary the strides in CNN from 1-5, the testing accuracy is highest at around 3. We use kernel size k, k+1 and k+2 for the 3 channels in our final framework.

As we vary k over 1-5, the testing accuracy first decreases for k = 2 and then increases. The maximum testing accuracy is observed when dropout = 0.30, decreasing on both sides of this point. As we increase the number of hidden states (units) of the GRU, the testing accuracy increases and then decreases. Our final F1 score of **0.9333** on Twitter15 dataset is when dropout is 0.30 and the batch size used for training is 40.

8 Conclusion

Our model AMRCN outperforms the baseline models and provides a reasonable explanation by highlighting the most important words/phrases towards the final prediction. We perform extensive experiments on two real-world datasets and observe the effectiveness of this attention-enhanced framework in generating explanations. Our model and baseline implementation and used datasets have been released [3].

References

Vidit Jain, Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang, and Yashvardhan Sharma. 2021. Aenet: an attention-enabled neural architecture for fake news detection using contextual features. *Neural Computing and Applications*, pages 1–12.

Yunfei Long. 2017. Fake news detection through multi-perspective speaker profiles. Association for Computational Linguistics.

Yi-Ju Lu and Cheng-Te Li. 2020. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. Association for Computational Linguistics.

Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. 2019. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*.

Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. 2018. Declare: Debunking fake news and false claims using evidence-aware deep learning. *arXiv preprint arXiv:1809.06416*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier.

[3] https://bit.ly/AMRCN_

Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A deep ensemble framework for fake news detection and classification. *arXiv preprint arXiv:1811.04670*.

Qiang Sheng, Xueyao Zhang, Juan Cao, and Lei Zhong. 2021. Integrating pattern-and fact-based fake news detection via model preference learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1640–1650.

Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 395–405.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

A Intuition Behind Lime

LIME is a local surrogate model, which means that it is a trained model used to approximate the predictions of the underlying black-box model. But, it comes with the idea to generate variations of the data into the machine learning model and tests what happens to the predictions, using this perturbated data as a training set instead of using the original training data.

In other words, LIME generates a new dataset consisting of permuted samples and the corresponding predictions of the black-box model. On this new dataset, LIME then trains an interpretable model (e.g., Lasso, decision tree, ...), which is weighted by the proximity of the sampled instances to the instance of interest.

A.1 Shortcomings of LIME

The LIME framework is wildly used nowadays. It has several significant problems, which make LIME a poor choice for the model interpretation, **(1)** Explanations of similar examples might be totally different, **(2)** LIME explanations fidelity is low, and **(3)** Explanation depends on the choice of LIME hyperparameters. To overcome these disadvantages we use the attention mechanism to highlight the most important words.