Neural-Network Models for Detecting Fake-News

Min Hun Lee, Sumeet Kumar

Carnegie Mellon University
Pittsburgh, PA 15213
minhunl, sumeetku @andrew.cmu.edu

Abstract

Misinformation on web, commonly known as fake-news, has become an evil in society and is impacting people in various ways. In this project, we try to identify statements (or news-stories) that have incorrect facts. Using LIAR dataset which has over 12,000 labeled statements, we design and evaluate many traditional and neural-network models. Our algorithms use extracted text-features such as Term Frequency-Inverse Document Frequency, and learns features from data using Neural-networks. Using Convolution Neural Network (CNN) and its hybrid variations that allow to use information like article's meta-data in addition to text, we slightly improve the performance of the LIAR dataset paper, but it is statistically not significant. We also tried to include fact-checking by using knowledge graphs (did not work!!), and by verifying information on Wikipedia. In particular, we use skip-thought embeddings of statements and Wikipedia text as additional input features to Hybrid CNNs. Despite trying many approaches, our results do not significantly exceed the results reported in the original LIAR dataset paper, but we are confident that our models are learning important patterns and the results can further improve by fine-tuning.

1 Introduction

'FDA announced that vaccines are causing autism.' was a popular statement circulated on social media. The statement was first found on 'truthcommand' website [1], but was later removed. The statement was read and shared by many on social-networking websites like Twitter. Later, reviewers on Politifact, a fact checking website, found the statement to be incorrect [2]. Many such false statements, are circulated on online social-networks every day. False statements or fake-news are not new. However, the phenomenal growth of social-media coupled with ease in publishing unverified content and click-based advertisement revenue, have made fake news a strong influencer in driving discussions [22]. Though often considered innocuous, such influences may have high social cost [4, 1]. For example, Parkinson [16] reported that the sheer inactivity of social media companies in stopping the spread of fake-news might have contributed to the results of the 2016 US presidential election. Further, the problem of false information spread becomes more acute when computers-agents are involved to rapidly spread them. Shao et al. [24] observed on Twitter during 2017 US presidential election to find that social bots are instrumental in spreading fake-news.

In the era of click-to-share and automated bots, traditional fact checking by journalists is not a way forward . The challenge has to be solved in an automated way to block the diffusion of misinformation as early as possible. Researchers have tried to use language features like 'sentiment' and 'length of

[1]http://truthcommand.com/2017/11/now-official-fda-announced-vaccines-causing-autism/

[2]http://www.politifact.com/punditfact/statements/2017/dec/19/truthcommandcom/no-fda-didnt-hide-information-linking-vaccine-auti/

headline' to identify fake news. However, such features are heavily dependent on training data, and do not generalize. well. Given that most fake stories have incorrect information, though difficult to delineate, we propose to use fact-checking against knowledge graphs and more trusted source of information like Wikipedia to detect fake-news.

In the midst of this increasing threat of fake-news and misinformation spread, the advancements in the artificial intelligence techniques provide a great opportunity. In particular, learning from the humongous amount of unstructured text data available on the web and creating knowledge graphs [7, 17, 2] offer a possibility of computational fact-checking of news. Hassan et al. [6] identified check-worthy factual claims in presidential debates, and Shi et al. [25] modeled the fact-checking problem as a link prediction task in a knowledge graph. Though automated fact checking for detecting fake-news is becoming an active area of research, most contributions are limited to small datasets with hand-labeled examples. For example, in [6], authors use a hand-labeled dataset of claims. In this project, our goal is to automate learning language features and fact-checking, so that they can be applied to any news-story or text document. In this research, we designed and evaluated many traditional and neural-network models. Our models use extracted text-features (such as term counts and Term Frequency-Inverse Document Frequency), and learns features from data using Neural-networks. Using Convolution Neural Network (CNN) and its hybrid variations that allow to use information like article's meta-data in addition to text, we get performance comparable to the LIAR dataset paper. We also included fact-checking by using knowledge graphs (this did not work!!) and verifying information on Wikipedia. We tried to improve knowledge-graphs by learning better embeddings that uses attributes (like political orientation) of nodes. However, since most entities (80%) in our dataset was found to be not present in 'DBpedia', we move this work to Appendix. Finally, we tried fact-checking by comparing statements embeddings and embeddings of sentences of statement's-speakers Wikipedia page.

The important contributions of this paper are:

- We built and evaluated many traditional machine learning models and neural-networks for learning representation to classify false-statements.
- Our Hybrid model extends CNNs by including meta-data and fact-checking features to improve performance over simpler models. In particular, we generate fact-check features by comparing statements and Wikipedia pages of speakers.
- We also tried fact-checking by using knowledge graphs and comparing entities and their relationships. However, since most entities in our dataset is not present in 'DBpedia', we move this work to Appendix. We tried to improve fact-checking on knowledge-graphs by learning better embeddings that uses attributes (like political orientation) of nodes.

The paper is outlined as follows. First, we discuss prior work in section 2. We then introduce our dataset in sec. 3. In section 4, we describe our models. We present our experiments and results in sec. 5. We finally conclude and discuss the future work.

2 Related Work

2.1 Fake News and Fact Checking

Rubin et. al. [21] demonstrate that the truthful and deceptive stories have different patterns in rhetorical structures using a sample of 36 self-rank personal stories. Rashkin et al. [18] discuss language characteristics of real news compared to satire, hoaxes and propaganda. They find that using Linguistic Inquiry and Word Count (LIWC) and sentiment lexicon features could be useful to understand the differences between fake and more reliable digital news sources. Jin et. al. [8] use topic models to identify opposing view points. They build a credibility network that links inter-tweet relations to verify the news posts in social networks.

Hassan et al. [6] categorize sentences in presidential debates into three categories (i.e. non-factual, unimportant, and check-worthy). They detected check-worthy factual claims in presidential debates using supervised algorithms (i.e. Multinomial Naive Bayes Classifier, Support Vector Classifier, and Random Forest Classifier). Shi et al. [25] modeled the fact-checking problem as a link prediction task in a knowledge graph. Ciampaglia et al. [3] used shortest paths between nodes in a knowledge graph for fact checking. In another similar work, Wu et al. [29] designed a framework to transform claims into queries and use efficient algorithms to measure not only the correctness of claims but also gauge the quality of claims. Though automated fact checking for detecting fake-news is becoming an active area of research, most contributions are limited to small datasets with hand-labeled examples.

2.2 Knowledge Extraction and Relational Learning

Text documents and webpages have been used for knowledge extraction [20]. However, knowledge extraction from news and social media posts are less common. Brambilla et al. [2] use Dandelion [3] to match texts into entities in social media and rank them to find a core entity type. Alternatively, we can apply relational machine learning that statistically analyzes relational or graph-structured data [15]. Specifically, neural network modules could be applied to relational learning. For example, Santoro et al. [23] tried neural-network for relational reasoning and were successful in learning entities and their relations in many tasks. It would be valuable to explore the effectiveness of using better embeddings of entities in knowledge graphs. Mittal et al. [14] recently presented the idea of unifying vector representation of entities and knowledge graphs. They tried their approach on a cyber-security informatics dataset. We propose extending this idea to knowledge graph entities in other domains (e.g. fake news detection).

Also Memory networks (MNs) could be tried to supplement knowledge graph models. Memory Networks [11, 26, 28], a neural network architecture with episodic memories, is becoming a tool of choice for solving question-answering problems. These memory networks can be trained end-to-end and have shown to get the state-of-the-art results on several types of question answering tasks like Facebook's bAbI dataset. The MNs are trained on raw input-question-answer triplets. During answering phase, these MNs use iterative attention process to search and retrieve relevant facts. We would like to explore if MNs can supplement knowledge graphs.

3 Dataset

We use LIAR dataset [27] for the task of detecting 'fake' news [4]. We extend the original dataset shared by Wang et. al. with additional data retrieved from Poltifact websites. The additional fields include 'source websites'. We parsed the source website URLs to include only the root URL. Table 1 describes the summarized statistics of dataset. This dataset includes 12,836 short statements, which are collected from various contexts, such as news release, TV, or radio interviews. Each statement has a human-labeled truthfulness score: *'pants-fire'*, *'false'*, *'barely true'*, *'half-true'*, *'mostly-true'*, *'true'*. Average length of statement is around 17.9 tokens. In addition to text statements, there are additional six fields of metadata information about statements: 1) subjects of a statement, 2) the speaker, 3) the job of a speaker, 4) the State information, and 5) the party affiliation. Finally, this dataset includes five numerical counts of the total credit history for each statement: 1) *'barely true'* counts, 2) *'false'* counts, 3) *'half-true'* counts, 4) *'mostly-true'* counts, and 5) *'pants-fire'* counts.

Table 1: LIAR Dataset Statistics

	Statistics
Training Set Size	10,269
Validation Set Size	1,284
Test Set Size	1,283
Avg. statement length (tokens)	17.9
Labels	6

4 Methodology

Our main goal is to learn language features for detecting fake-news. The following are main tasks in this project:

- Extract text features from statements

[3]https://dandelion.eu/

[4]http://www.cs.ucsb.edu/ william/data/liar_dataset.zip

- Compare different options (embeddings) for representing key units in text such as entities and concept
- Fine-tune pre-trained embeddings to adopt to target domain
- Learn joint representations based on text embeddings and text-metadata (like speaker, party inclination)
- Include fact-checking features by using external knowledge-graphs or other trusted source of information (like Wikipedia)
- Experiment with different Neural-Networks by varying the size of layers and other parameters.

4.1 Loss function and Baseline Models

$$\mathscr{L}(y,p) = -\frac{1}{n}\sum_{i,j} y_{ij}\log(p_{ij}) \tag{1}$$

For the classification task, we use two metrics 'accuracy' and 'cross-entropy'. For training neual-networks, we define our overall loss function using cross-entropy loss as can be seen in Equation 1, where $i \in n$ samples, $j \in m$ classes, y is the (one-hot) true label, p is the prediction label.

Our baseline models include both traditional machine-learning algorithms and Neural Network models. We build multiple traditional machine-learning algorithms, such as 'multinomial Naive Bayes', 'Random Forest', 'Support Vector Machine (SVM)'. These traditional machine learning algorithms utilize numerical statistics of words: term counts and Term Frequency-Inverse Document Frequency (TF-IDF).

For Neural Network models, we consider a Convolutional Neural Network (CNN) with an unsupervising neural language model, which has recently shown promising results in sentence classification [9]. This baseline CNN model utilizes only text statement to detect fake news. The baseline CNN model utilizes pre-trained word embeddings from skip-gram and Continuous Bag of Words (CBOW) models of Gensim's Word2Vec library [19]. The baseline CNN model includes a single filter with the following layers in the network: an embedding layer, a 'Dropout' layer(25%), a 'MaxPooling1D' later, a 'Flatten' layer, a 'Dense' layer, a 'Dropout' layer, and finally an 'Activation' layer.

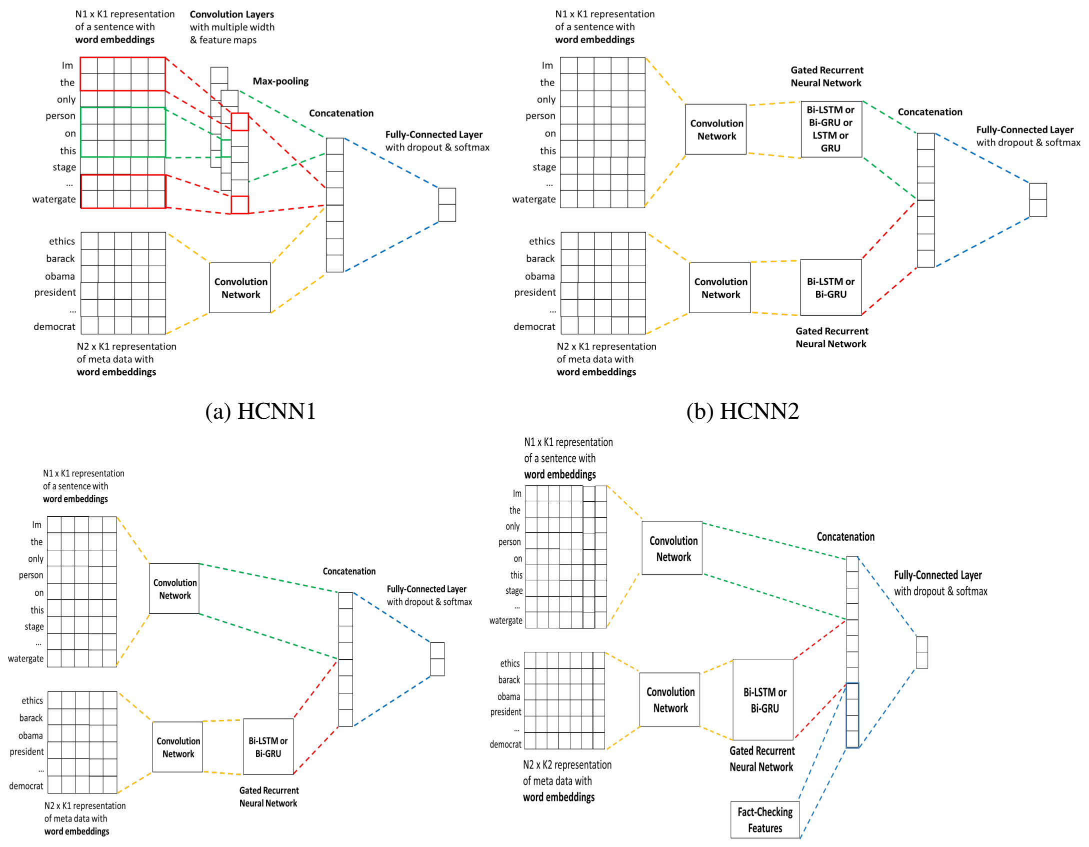
4.2 Hybrid Convolutional Neural Networks with Text and Meta-Data

To improve the baseline CNN model, we design Hybrid Convolutional Neural Networks (HCNNs) that utilize both text and meta-data of a statement. Figure 1 describes the overall architectures of models. A HCNN employs two CNNs: one for text data and the other for meta-data. The model architecture of each CNN in HCNN model follows the architecture of [9]. We apply different length of a sentence for padding text and meta-data. In addition, each CNN has independent embedding matrix as shown in Figure 1(a).

For word embedding matrix, we explore both random initialization and pre-training with Gensim's word2vec API. As stated in [9], we consider the following model variations: 1) HCNN-rand, 2) HCNN-static, and 3) HCNN-nonstatic. HCNN-rand represents a hybrid model that randomly initializes embedding and learns during training. HCNN-static loads pre-trained vectors from word2vec and randomly initializes unknown words. It only learns the parameters of the model except word embedding matrix. HCNN-nonstatic is the same as HCNN-static, but it tunes embedding matrix during training.

After representing a input sentence with word embeddings, both CNNs employ 50% drop-out and the identical operations as follows: convolutions with multiple filters, max-pooling. After max-pooling layer at each CNN, a HCNN model concatenates max-pooled text outputs. This model employs another drop-out layer (80%) for the regularization and passes them to a fully connected softmax layer. A HCNN model utilizes the output of the softmax activation to detect a fake news. Unlike the baseline CNN model, each CNN of a HCNN model utilizes multiple filters to have richer representations of fake news. We refer a hybrid model with two CNNs as HCNN1.

HCNN2 has two CNNs that have the same structures with those of HCNN1. The difference is that two CNNs applies the gated Recurrent Neural Network (RNN) models before concatenation as shown in Figure 1(b). For the gated RNNs, we explore both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). We also utilize bidirectional sequential flows, which are useful in the speech recognition domain [5]. The use of bi-directional sequences provide not only past information (e.g. the previous words), but also future information (e.g. the next words) to build up the global context of a sentence.


N1 x K1 representation of a sentence with word embeddings
Convolution Layers with multiple width & feature maps
Max-pooling
Concatenation
Fully-Connected Layer with dropout & softmax
Im
the
only
person
on
this
stage
...
watergate
ethics
barack
obama
president
...
democrat
Convolution Network
N2 x K1 representation of meta data with word embeddings
Gated Recurrent Neural Network
Bi-LSTM or Bi-GRU or LSTM or GRU
Bi-LSTM or Bi-GRU
N2 x K2 representation of meta data with word embeddings
Fact-Checking Features

(a) HCNN1 (b) HCNN2

(c) HCNN3 (d) HCNN4

Figure 1: (a) HCNN1 merges text-based CNN and meta-based CNN. (b) HCNN2 includes two Gated CNN models. (c) HCNN3 contains text-based CNN and Gated CNN for meta-data (d) HCNN4 contains all inputs of HCNN3, and includes additional features from the fact-checking approach.

Meta-data is summarized entity representations of a sentence. we consider applying bi-directional flows to meta-data might be useful to encode a global context representation of being a fake or not. However, it might not be useful for texts of a sentence, because some words are not related to represent a sentence nor be a true statement. In [27], Wang described that the application of Bi-LSTM on text data for a fake news detection leads to a over-fitted model. We wonder whether unidirectional LSTM might lead to useful representations. Thus, we consider both Bi-LSTM and LSTM on a CNN model for text data. As the application of LSTM models might not be helpful for texts of a sentence, the model architecture of HCNN3 applies only gated RNN on a CNN model for meta-data as shown in Figure 1(c). HCNN4 contains all inputs of HCNN3, and includes additional features from the fact-checking approach.

For HCNN models, we also apply a feature engineering to explore the usefulness of weighted word embeddings as shown in Figure 4. Given the meta-data information (e.g. history of saying false statement, a subject), we create a weight of each statement. We then multiply weights with word embedding inputs to generate weighted input word embeddings. For this project, we mainly utilized negated and normalized counts of being false as a weight vector. If there is no history false statement, a weight has the value of one to represent original word embedding inputs. Only words that are related to false statement will become multiplied negative weight values. The goal of the negation is to explore whether this multiplication of negative weights can be useful to distinguish deceitful words and trustworthy words.

4.3 Including Knowledge Graphs and Wikipedia Articles for Fact-Checking

It is good to learn text and meta-data features for detecting fake statements. However, a statement is false, even if does not have any textual features e.g. 'Sun revolves around earth' is a false statement irrespective of absence of any obvious language features. This leads us to look for incorrect factual content is statements which cannot be easily learned by neural-networks. We could think to two such approaches: a) Spiting a statement into entities and relations, and verifying the presence of such relations against a dataset like 'DBpedia'. b) Fact-checking by searching statements in other trusted sources like Wikipedia. We tried both the approaches which are described next.

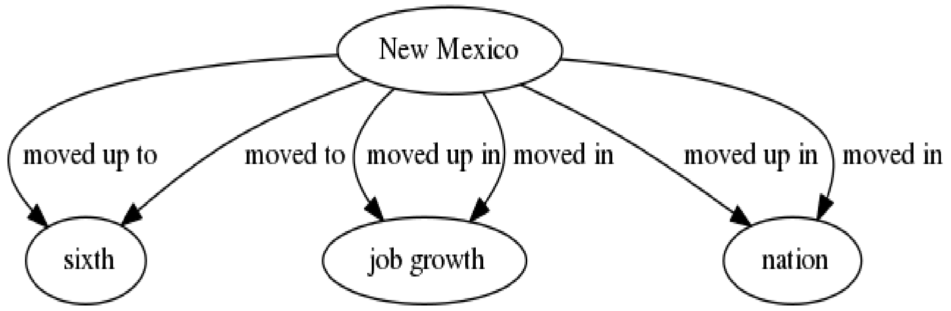


Figure 2: Entities and Relations extraction using Stanford NLP library pipeline. For example, we used a statement 'New Mexico moved up to sixth in the nation in job growth.' to obtain one 'Subject' and six possible relations. Note that the relations extractor is not perfect.

In approach a, we extract knowledge from statements as entities and relations. We first use 'Stanford NLP parser' and 'Stanford Open Information Extraction' libraries [12] to retrieve 'entities' as 'subjects' and 'objects', and their 'relationships'. Once we have retrieved all possible entities from statements (see Fig. 2), we use 'DBpedia' [5] knowledge graph to verify if these entities and relations exist. 'Dbpedia' project is derived from Wikipedia, and allows to query relationships of Wikipedia entities. We tried to find the entities in our dataset (statements) in 'DBpedia', and found that only 20.4% of entities are present in this dataset. Given this small fraction of entities, it was very unlikely to have links between these entities. Thus, the straightforward approach to verify relations between entities using 'DBpedia' dataset is not feasible. We tried to use entities embedding in place of 'entities' as text, but this experiment could not be completed on time, so we have moved it to Appendix.

We then consider an alternate approach for fact-checking using Wikipedia pages. Though the total number of entities ($\sim$ 30,000) in our dataset is large, the number of speakers (who issued statements) in our dataset is relativels small (3,310). Most of these speakers (78%) have a Wikipedia entry. Also the statements in the LIAR datasets are important statements as they were picked by Politifact for fact-checking. If the statements are true, it is possible that such statements are also included on Wikipedia. Following this idea, we retrieve Wikipedia articles of all speakers in the LIAR dataset. We then try to search the statements given by a speaker with sentences on Wikipedia for that particular speaker. Given that the 'statements' may not a have a direct match on Wikipedia, but possibly a similar meaning, We use skip-thought [10] sentence embedding to encode the LIAR dataset statements. We also encode all sentences in Wikipedia articles on speakers as skip-thought embeddings, and then use a similarity measure between the statement and the Wikipedia sentences. Finally, we take top n (a parameter) Wikipedia sentences closest to the Politifact statement, based on their embedding similarity measure. We create n-size vector with distance measure between the statement and the n closest sentences. This vector is fed to a Hybrid CNN that uses this similarity vector as one of the input features. Figure 3 explains the core idea used in this approach. Hybrid Model HCNN 4 uses the fact checking features as shown in Fig. 1.

5 Experiments & Results

In this section, we describe our implementation and model specification. For traditional machine learning algorithms, we use Python 'sklearn' library and a grid-search for optimizing model parameters. For Neural Network models, we used 'keras' library with Tensorflow backend. For the baseline

[5]https://en.wikipedia.org/wiki/DBpedia

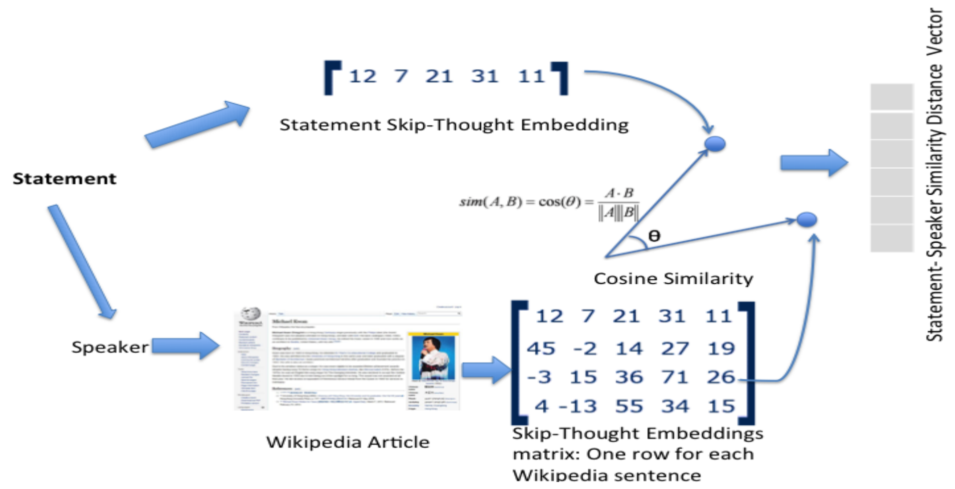

12 7 21 31 11
Statement Skip-Thought Embedding
Statement
sim(A, B) = cos(θ) = A·B / ‖A‖‖B‖
θ
Cosine Similarity
Statement- Speaker Similarity Distance Vector
Speaker
Wikipedia Article
12 7 21 31 11
45 -2 14 27 19
-3 15 36 71 26
4 -13 55 34 15
Skip-Thought Embeddings matrix: One row for each Wikipedia sentence

Figure 3: Fact-checking Approach: For fact-checking, we compare a 'statement' and the statement author's wikipedia page. The assumption is that if a statement spoken by someone is important and trustworthy, it is likely to find a mention on Wikipedia. Rather than directly comparing the statement and Wikipedia-text, we compare skip-thought [10] encoding of statements and text on Wikipedia. We finally use cosine-similarity to select the top few Wikipedia-sentence embedding, and generate a similarity-distance vector that is fed to a Hybrid CNN as an input.

Table 2: Performance on models on validation dataset. 'Random Forest' model utilizes only text data and achieves a comparable performance to the baseline CNN model of the baseline paper. The hybrid models use both text and meta-data features. More details on HCNN models for different parameters is available in Appendix.

	Performance on Validation and Test Set	
Models	Validation-Accuracy	Test-Accuracy
HCNN1 (best result among various model selections)	0.2764	0.2454
HCNN2 (best result among various model selections)	0.2632	**0.2659**
HCNN3 (best result among various model selections)	**0.2858**	0.2628
HCNN4 (best result among various model selections)	0.2791	0.2532
HCNN3 after fine-tuning with Text + Speaker meta-data	0.2718	0.2667
HCNN3 after fine-tuning-1 with Text + All meta-data	0.2710	**0.2778**
HCNN3 after fine-tuning-2 with Text + All meta-data	**0.2897**	0.2636
CNN	0.2414	0.1569
KNN	0.2310	0.2121
SVM-L1	0.2380	0.2309
BernouliNB	0.2360	0.2341
Perceptron	0.2070	0.2003
ElasticNet	0.2440	0.2359
RandomForest	0.2590	0.2330
Baseline paper [27] - CNN with Text data	0.260	0.270
Baseline paper [27] - Hybrid CNN Text + Speaker meta-data	**0.277**	0.248
Baseline paper [27] - Hybrid CNN Text + All meta-data	0.247	**0.274**

CNN model, we use 'early stopping 'dropout' as regularization techniques. The model used 'SGD' optimization, with a 'Categorical Cross-Entropy' loss function. The other parameters are: lr=0.01, decay=1e-6, momentum=0.9. Similar to the result of [27], LSTM with text data become over-fitted and has worse performance than this CNN model, so we don't include in the results.

Table 3 shows various models and their empirical results on validation and test dataset. The results reported are for the best found parameters. Overall, the accuracies of baseline models range between 0.2070 and 0.2590. The performance of 'Random Forest' model has the best performance among the baseline models, which is comparable to the performance of the baseline CNN model in [27]. For the training our proposed HCNN, we analyze the trends of train and validation loss during the training (Figure 5 in the Appendix). We then decide to use 5 Epochs over various model selections. Our proposed HCNN models perform mostly around 0.2352 and 0.2620. As the performance was not close the best results of the baseline paper [27], we changed the filter size of CNNs from (3, 4, 5) to (3, 8) as described in [27]. The experimental results with (3, 8) filter sizes have included in the Appendix. We only included the best result of each Hybrid CNN model in the Table 3. After changing the filter sizes, we are able to achieve the results comparable to the baseline paper [27]. Specifically, HCNN3-Rand Bi-GRU for Meta-data with weighted embedding inputs model performs the best among various model options in the Appendix. Given this model, we fined-tuned model by increasing another hidden layer in the fully connected layer (a single layer with 100 hidden units to 100 and 50 hidden layers) and utilize 100 billion words of Google News [13]. Our results of this model have been slightly improved as shown in the Table 3. Compared to the LIAR dataset [27], our model improves around 1.2 %.

5.1 Analysis and Discussion

To analyze the significance of the accuracy improvement using HCNNs over baselines, we conduct one-sample t-test with inputs as validation and test accuracies. First, we compare the 'HCNN3 after fine-tuning-1' model with RandomForest model that achieve the best performances among baseline models. The null hypothesis is that 'HCNN3 after fining-tuning-1' and RandomForest are equally good. For one-tailed hypothesis, the p-value is 0.000519, which implies the result is significant at $p < 0.05$ and $p < 0.01$. Thus, we conclude 'HCNN3 after fine tuning-1' model performs better than the baseline model, RandomForest. However, when we compare our Hybrid model with the baseline paper, the significance test shows that the performance improvement is not large enough to conclude outperforming the baseline liar paper.

Even if the improvement is not significant, our proposed hybrid model can learn meaning representation of a sentence to detect a false statement. One interesting aspect is that 'HCNN-3 BiGRU for Meta data with weighted embedding inputs' model outperforms other model selections. This implies that simple weighted embedding representation might lead to more meaning representation of a specific task. In this paper, we only had a chance to utilize negated normalized counts of being false. It would be interesting to explore whether this weighted embedding approach can be derived from the fact-checking features. We can easily analyze the effectiveness of using fact-checking features based weighted embedding input by comparing the result of this paper. If we notice any positive improvement, we would like to further experiment which approaches are useful.

This paper includes multiple model selection (e.g. Bi-GRU or Bi-LSTM). However, the overall performance difference is statistically not significant to derive any conclusion. We did not mention this conclusive comparison in this report.

6 Conclusions & Future Work

In this project, we built many machine-learning models to detect fake statements. We tried CNNs and Hybrid-CNNs that use meta-data and fact checking features, in addition to textual information in statements. For fact-checking, we tried knowledge graphs and more trusted information sources like Wikipedia. 'DBpedia' as the source of knowledge graphs was not very useful as most entities in our dataset were not present in this dataset. This was possibly because we considered exact match of entities name. We tried to use learn embedding vectors for entities, but this work could not be completed on time, and so moved to Appendix for future consideration. For fact-checking, we tried another approach of comparing statements against Wikipedia entries of statement speaker (or author). Despite trying many approaches, our test-results do not (statistically) exceed the results reported in the original LIAR dataset paper, but we are confident that our models are learning important patterns and the results can further improve by fine-tuning.

In future, we would try to fine tune our models by trying different parameter settings. We will also try checking against a larger knowledge-graphs. We attempt to use knowledge-graph failed because of sparsity of entities in 'DBpedia'. We tried to resolve it use node-embeddings, but this approach could not be completed on time for this report. We will continue to work on designing better node-embeddings. One idea is to explore the possibility to derive and learn multiple weighted embeddings together.

References

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2) 211-36., 2017.

[2] M. Brambilla, S. Ceri, E. Della Valle, R. Volonterio, and F. X. Acero Salazar. Extracting emerging knowledge from social media. In *Proceedings of the 26th International Conference on World Wide Web*, pages 795–804. International World Wide Web Conferences Steering Committee, 2017.

[3] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015.

[4] E. Ferrara. Manipulation and abuse on social media. *ACM SIGWEB Newsletter*, (Spring):4, 2015.

[5] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[6] N. Hassan, C. Li, and M. Tremayne. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1835–1838. ACM, 2015.

[7] L. Heck, D. Hakkani-Tür, and G. Tur. Leveraging knowledge graphs for web-scale unsupervised semantic parsing. 2013.

[8] Z. Jin, J. Cao, Y. Zhang, and J. Luo. News verification by exploiting conflicting social viewpoints in microblogs. In *AAAI*, pages 2972–2978, 2016.

[9] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[10] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[11] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387, 2016.

[12] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[13] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[14] S. Mittal, A. Joshi, and T. Finin. Thinking fast, thinking slow! combining knowledge graphs and vector spaces. *arXiv preprint arXiv:1708.03310*, 2017.

[15] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[16] H. J. Parkinson. Click and elect: how fake news helped donald trump win a real election. *The Guardian*, 2016.

[17] J. Pujara, H. Miao, L. Getoor, and W. W. Cohen. Using semantics and statistics to turn data into knowledge. *AI Magazine*, 36(1):65–74, 2015.

[18] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2921–2927, 2017.

[19] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[20] X. Ren, Z. Wu, W. He, M. Qu, C. R. Voss, H. Ji, T. F. Abdelzaher, and J. Han. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conferences Steering Committee, 2017.

[21] V. L. Rubin and T. Lukoianova. Truth and deception at the rhetorical structure level. *Journal of the Association for Information Science and Technology*, 66(5):905–917, 2015.

[22] S. Samanth. Inside the macedonian fake-news complex. *Wired.com*, 2017.

[23] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

[24] C. Shao, G. L. Ciampaglia, O. Varol, A. Flammini, and F. Menczer. The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, 2017.

[25] B. Shi and T. Weninger. Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102. International World Wide Web Conferences Steering Committee, 2016.

[26] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[27] W. Y. Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[28] J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[29] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600, 2014.

7 Appendix

7.1 Feature Engineering: Weighted Embedding Matrix

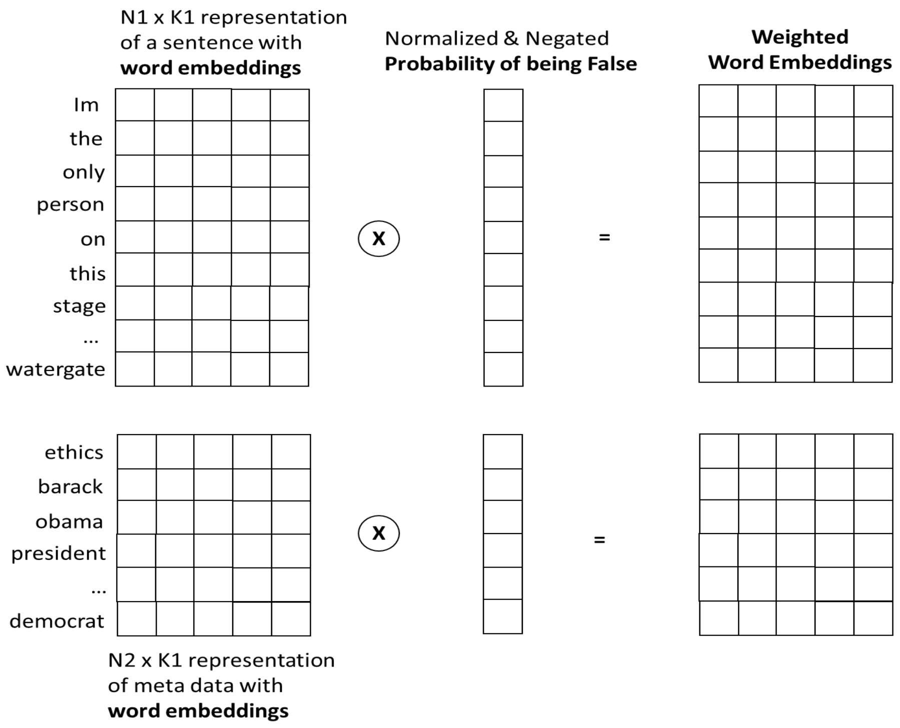

N1 x K1 representation of a sentence with **word embeddings**
Normalized & Negated **Probability of being False**
Weighted Word Embeddings
Im
the
only
person
on
this
stage
...
watergate
X
=
ethics
barack
obama
president
...
democrat
X
=
N2 x K1 representation of meta data with **word embeddings**

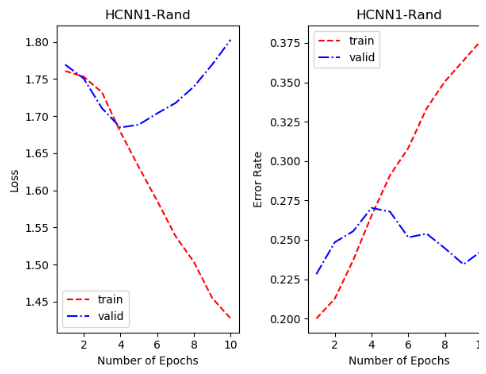
Figure 4: Process of generating weighted word embeddings. First, we compute normalized counts of credit history. We then negate the normalized false count of each statement to create a vector that represents negative probability of being false statement. Given input sentence representation with word embeddings, we compute element-wise multiplication between input word embedding and a probability vector to generate weighted word embeddings

7.2 Baseline Results in the Midway Report

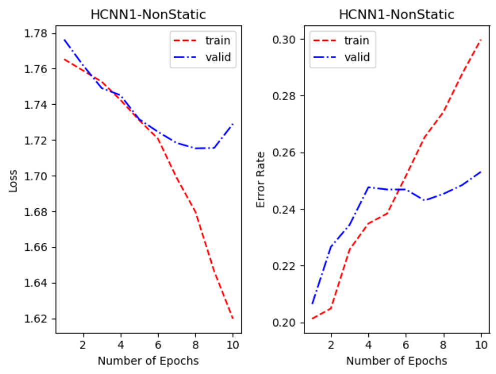
Table 3: Performance on models on validation dataset. 'Random Forest' model utilizes only text data and achieves a comparable performance to the baseline CNN model of the baseline paper. The hybrid CNN model of the baseline paper uses both text and meta data.

	Performance on Validation Set			
Models	Accuracy	F-Score	Precision	Recall
CNN	0.2414	0.1569	0.1692	0.2044
KNN	0.2310	0.2121	0.2309	0.2126
SVM-L2	0.2320	0.2274	0.2334	0.2253
SVM-L1	0.2380	0.2309	0.2381	0.2294
NaiveBayes	0.2270	0.2242	0.2443	0.2189
BernouliNB	0.2360	0.2341	0.2387	0.2318
Perceptron	0.2070	0.2003	0.2030	0.1993
ElasticNet	0.2440	0.2359	0.2385	0.2354
RandomForest	0.2590	0.2330	0.2947	0.2350
Baseline paper [27] - CNN	0.260			
Baseline paper [27] - Hybrid CNN Text + Speaker	0.277			

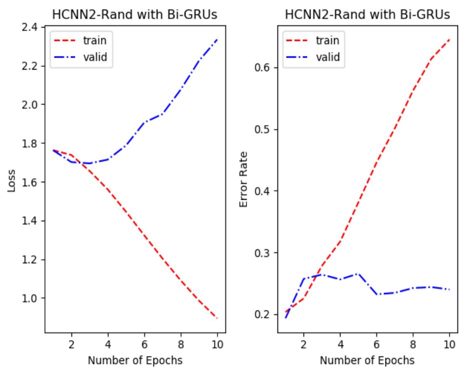
7.3 Experiment Results of HCNNs



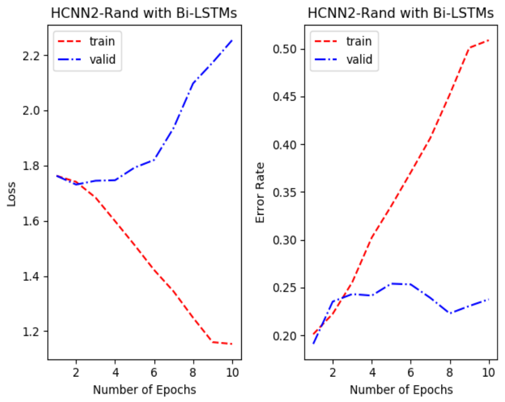
(a) HCNN1-Rand



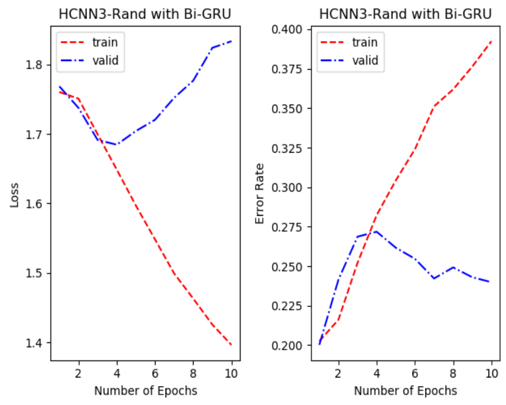
(b) HCNN1-NonStatic



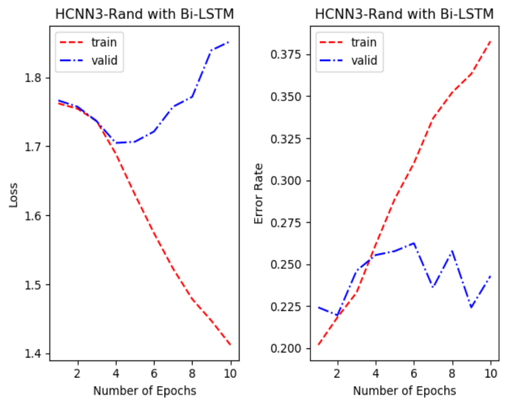
(c) HCNN2-Rand Bi-GRUs



(d) HCNN2-Rand Bi-LSTMs



(e) HCNN3-Rand Bi-GRU for meta-data



(f) HCNN3-Rand Bi-LSTM for meta-data

Figure 5: Plots show the trends of loss and accuracies during the training phase with train and validation dataset. For each model, we can identify that both train and validation loss decreases until a certain point, which implies our network is learning some meaningful features. To select optimal Epochs of the experiment, we run at least 10 Epochs to check the trends of over-fitting where train loss decreases, but validation loss increases. Each Hybrid Model has slightly different value of Epoches to become over-fitted. Overall, we easily notice a over-fitting from 5-6 Epochs. Thus, we conduct our experiment mostly using 5 Epochs.

Table 4: Results of HCNN1 with using all meta-data

HCNN1	Train	Valid	Test
Rand	0.3965	0.2764	0.2454
Static	0.2876	0.2733	0.2707
NonStatic	0.2585	0.2562	0.2454

Table 5: Results of HCNN1 with using only speaker meta-data

HCNN1	Train	Valid	Test
Rand	0.5170	0.2647	0.2620
Static	0.2414	0.2437	0.2383
NonStatic	0.3220	0.2492	0.2525

Table 6: Results of HCNN2 with various model selections using all meta-data

HCNN2	Model Selection	Train	Valid	Test
Rand	2 Bi-LSTMs	0.425	0.2445	0.2367
	2 Bi-LSTMs with Weighted Embeddings	0.4254	0.2538	0.2209
	LSTM for Text Bi-LSTM for Meta	0.2596	0.2336	0.2446
	LSTM for Text Bi-LSTM for Meta with Weighted Embeddings	0.2539	0.2429	0.2407
	2 Bi-GRUs	0.5207	**0.2609**	0.2525
	2 Bi-GRUs with Weighted Embeddings	0.4226	0.2352	0.2202
	BiGRU for Meta GRU for Text	0.4662	0.2570	0.2573
	BiGRU for Meta GRU for Text with Weigthed Embeddings	0.4421	0.2523	0.2494
NonStatic	**2 Bi-LSTMs**	0.3166	0.2507	**0.2596**
	2 Bi-LSTMs with Weighted Embeddings	0.3270	0.2593	0.2525
	LSTM for Text Bi-LSTM for Meta	0.2274	0.2289	0.2391
	LSTM for Text Bi-LSTM for Meta with Weighted Embeddings	0.2264	0.2258	0.2194
	2 Bi-GRUs	0.3778	0.2352	0.2533
	2 Bi-GRUs with Weighted Embeddings	0.3903	0.2359	0.2517
	BiGRU for Meta GRU for Text	0.3617	0.2554	0.2438
	BiGRU for Meta GRU for Text with Weighted Embeddings	0.3661	0.2445	0.2509

Table 7: Results of HCNN2 with various model selections using only speaker meta-data

HCNN2	Model Selection	Train	Valid	Test
Rand	2 Bi-LSTMs	0.3624	0.2383	0.2154
	2 Bi-LSTMs with Weighted Embeddings	0.3827	0.2515	0.2328
	LSTM for Text Bi-LSTM for Meta	0.2648	0.2313	0.2280
	LSTM for Text Bi-LSTM for Meta with Weighted Embeddings	0.2371	0.2110	0.2241
	2 Bi-GRUs	0.5642	0.2554	**0.2659**
	2 Bi-GRUs with Weighted Embeddings	0.5256	**0.2632**	0.2391
	BiGRU for Meta GRU for Text	0.4715	0.2601	0.2359
	BiGRU for Meta GRU for Text with Weigthed Embeddings	0.4715	0.2601	0.2359
NonStatic	2 Bi-LSTMs	0.3958	0.2492	0.2557
	2 Bi-LSTMs with Weighted Embeddings	0.2908	0.1861	0.2138
	LSTM for Text Bi-LSTM for Meta	0.1916	0.1954	0.1902
	LSTM for Text Bi-LSTM for Meta with Weighted Embeddings	0.2064	0.1931	0.2091
	2 Bi-GRUs	0.3559	0.2328	0.2344
	2 Bi-GRUs with Weighted Embeddings	0.3360	0.2476	0.2359
	BiGRU for Meta GRU for Text	0.4715	0.2601	0.2359
	BiGRU for Meta GRU for Text with Weighted Embeddings	0.3691	0.2390	0.2423

Table 8: Results of HCNN3 with various model selections using all meta-data

HCNN3	Model Selection	Train	Valid	Test
Rand	Bi-LSTM for Meta	0.3624	0.2383	0.2154
	BiLSTM for Meta with Weighted Embeddings	0.3749	0.2788	0.2604
	Bi-GRU for Meta	0.3815	0.2679	0.2596
	Bi-GRU for Meta with Weighted Embeddings	0.3954	0.2507	0.2565
NonStatic	BiLSTM for Meta	0.2852	0.2834	0.2478
	BiLSTM for Meta with Weighted Embeddings	0.2673	0.2679	0.2470
	Bi-GRU for Meta	0.29003	0.2546	0.2470
	Bi-GRU for Meta with Weighted Embeddings	0.2972	**0.2858**	**0.2628**

Table 9: Results of HCNN2 with various model selections using only speaker meta-data

HCNN3	Model Selection	Train	Valid	Test
Rand	Bi-LSTM for Meta	0.3642	0.2437	0.2486
	BiLSTM for Meta with Weighted Embeddings	0.3829	0.2577	0.2391
	Bi-GRU for Meta	**0.3694**	**0.2655**	**0.2667**
	Bi-GRU for Meta with Weighted Embeddings	0.3644	0.2655	0.2478
NonStatic	BiLSTM for Meta	0.2795	0.2546	0.2438
	BiLSTM for Meta with Weighted Embeddings	0.2835	0.2554	0.2288
	Bi-GRU for Meta	0.2649	0.2538	0.2454
	Bi-GRU for Meta with Weighted Embeddings	0.2858	0.2616	0.24230