You Can't Handle the Truth

Tessa Patapoutian tpatapoutian@umass.edu
William Howe whowe@umass.edu
Zhixiang Huo zhuo@umass.edu
Nathan Xuan jxuan@umass.edu

1 Problem statement

We have worked on the problem of mitigating misinformation by implementing a fact-checking classifier which classifies claims based on their level of truthfulness. This is an important topic to explore because many people (e.g. politicians, news sources) make a plethora of claims, and most people can be overwhelmed by the amount of information and do not have enough time to fact-check every single claim. The information age is full of misinformation which can lead to anything from interpersonal to political strife. If the fact-checking process could be automated, then it would be faster and easier to verify important claims. Our measurement of truthfulness uses textual entailment as well as sentiment analysis and the semantic value of a statement, which is an often complex and subjective topic. We present several fact-checking classifiers which take either (1) a claim, (2) a claim and a justification, (3) a claim, metadata, and feature vector, or (4) a claim, justification, metadata, and feature vector. The models label the claims as either true or false in a binary classification task, and also label them on a 6-point truthfulness scale. We also investigate which of the 4 inputs (claim, justification, metadata, and feature vector) are the most useful in the classification task.

2 What you proposed vs. what you accomplished

- ~~Collect and preprocess dataset~~
- *Build and train baseline model which always predicts the most common truthfulness label from the training data*: We decided that this model was too simple
- ~~Build and train baseline model which uses sentence GloVe embeddings of each claim as the input to a single-layer NN classifier~~
- *Build and train a Logistic Regression model like that in Alhindi et al. (1) which uses a claim, justification, metadata, and semantic features*: We decided that this was model was too simple
- *Build and train a model with a dual BiLSTM architecture like that in Alhindi et al. (1) which uses a claim, justification, metadata, and semantic features*: We had trouble implementing a dual architecture so we changed our approach
- ~~Build and train a BiLSTM model like that in Alhindi et al. (1) which uses a claim, justification, metadata, and semantic features, and try to get similar results~~
- *Examine to what extent a shift in domain of the training and testing data affects the performance of the model*: We couldn't find the data to do this
- *Augment our model by adding extra features and an extra attention layer*: We decided to augment our model with BERT instead
- ~~Build and train a model using AllenNLP~~
- ~~Perform in-depth error analysis to figure out what kinds of examples our approach struggles with~~

3 Related work

The increased demand for automated fact-checking has meant that there have been much recent work done in this domain. Approaches vary in how exactly they define the task of fact-checking, specifically in how they define the input and output, as well as where they source their data.

Regarding input, using subject-predicate-object triples (e.g. *[Obama, graduated from, Harvard]*) has been popular because they are easy to fact check with structured knowledge bases (6). However, this requires a lot of processing to convert claims into these triples. Other work uses textual claims as input, such as Zhang et al. (13) who consider both directly and indirectly asserted claims to evaluate the trustworthiness of information sources. Their evaluation of claims is similar to our approach because they use textual entailment to assess whether or not given textual evidence entails a given claim. There are several reliable databases with fact-checked claims, which makes this input format advantageous. Claims can be fact-checked against knowledge bases or against textual sources. And other previous work has used whole documents as input, which means that claims must first be extracted as triples (11), or as sentences before they are fact-checked (14).

Regarding output, most fact-checking models label a claim as true or false as a binary classification task. Some label as true, false, and unknown (3) or another third label to signify neither support nor refute (10). Fact-checking databases like Politifact use a 6-point truthfulness scale, so models that use this data, such as ours and Alhindi et al. (1), often find it easy to use this same scale. Other approaches score claims instead of labeling them, as in Bast et al. (2). In general, the way that claims are labelled is varied and not always consistent, since truthfulness is an often subjective metric.

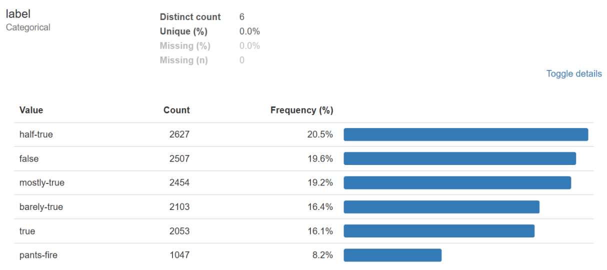
Regarding sources of evidence, some approaches use no evidence at all other than the claim (7). Others, like Wang (12), use metadata to classify truthfulness, like the source of the claim or the media source which the claim is presented in. Wang (2017) used a variety of models, such as logistic regression, SVMs, CNNs, and BiLSTMs, that integrated speaker-related metadata to see which metadata improved the performance of fake news detection the most. Our approach uses the dataset created in Wang (2017) and our model makes use of some of this metadata, especially the prior history of the speaker. Many approaches use knowledge graphs to store structured information about the world in a machine readable format, such as in Thorne and Vlachos (9). Hassan et al. (4) uses a variety of sources to fact-check claims, including Politifact.com, CNN, and FactCheck.org.

Our approach uses textual claims as input, a 6-way truthfulness classification, and the LIAR-PLUS dataset which was assembled from Politifact.com and contains textual claims, justifications, and metadata. Our model was largely inspired by the work done by Alhindi et al. (1), who explored what types of metadata would be important for prediction. They used multiple models, such as logistic regression, state vector machine, and bidirectional recurrent neural network. We decided to extend this work by implementing a similar BiLSTM model and exploring how the use of different metadata and embeddings affected performance.

4 Your dataset

Our data is challenging because it is almost impossible to have a completely unbiased annotated dataset of fact-checked claims, and there is also not widespread agreement on how fact-checked data should be annotated. The ground truth itself is sometimes hard to define, and truthfulness can often be a subjective metric. There could be different perspectives involved so that people disagree on the veracity of a claim, or there could be unclear evidence so the veracity can't be properly evaluated. If there are more than two possible labels in a dataset, then there are different levels of truthfulness that a claim can be labelled as, which makes the classification task even more subjective and open to human error. Also, different datasets annotate their data in different ways, such as labels vs scoring, number of labels, metadata, etc. Data in the fact-checking domain is not always 100% trustworthy or consistent, which makes this a difficult and interesting task.

Our model uses the LIAR-PLUS dataset for its training, validation, and testing. This dataset was assembled in 2018 using claims extracted from Politifact.com. There are 12,792 samples, where each includes a textual claim, metadata about the speaker and claim, a textual justification, and a truthfulness label. The metadata includes speaker information such as party affiliation (e.g. Democrat), speaker job title (e.g. Senator), and state (e.g. Massachusetts). The possible labels are ”true”, ”mostly-true”, ”half-true”, ”barely-true”, ”false”, and ”pants-fire”. 20.5% of the samples are labelled ”half-true”, 19.6% are labelled ”false”, 19.2% are labelled ”mostly-true”, 16.4% are labelled ”barely-true”, 16.1% are labelled ”true”, and 8.2% are labelled ”pants-fire”.



label
Categorical

Distinct count	6
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Toggle details

Value	Count	Frequency (%)
half-true	2627	20.5%
false	2507	19.6%
mostly-true	2454	19.2%
barely-true	2103	16.4%
true	2053	16.1%
pants-fire	1047	8.2%

Figure 1: The LIAR-PLUS dataset has 6 categories of veracity

One type of metadata was the prior history of the speaker, which tracks the labels of the speaker's previous statements. It could be imagined that speakers each have a different level of past veracity, some who are almost always truthful and some who are almost always misrepresenting reality, but this is not the case. From Figure 2, we can see that speakers tend to have a relatively even distribution between the different levels of truthfulness, although the ’pants-fire’ label seems to be an exception. This could mean that the prior history metadata does not have much predictive power.

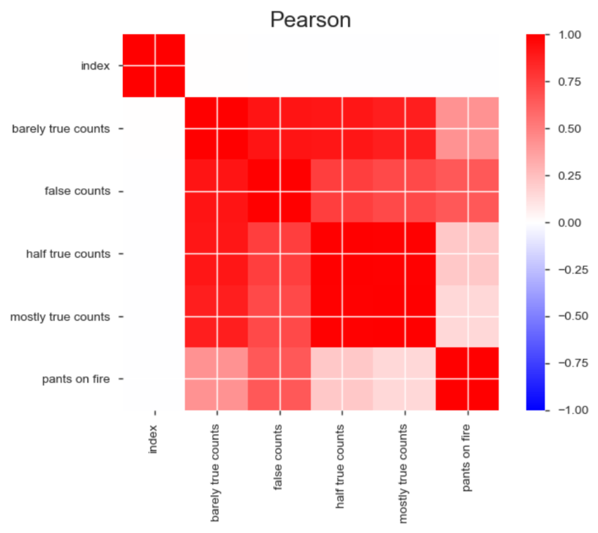


Figure 2: Correlations between past label histories of speakers

LIAR-PLUS entries are nearly complete. Each sample had a claim and a truthfulness label, although a small percentage of samples lacked a textual justification. Samples without justifications were excluded from the data in the preprocessing stage. As expected, the statement and justification features each have high cardinality.

Feature	Percent Unique	Percent Missing
Statement	99.8	0.0
Context	40.2	1.0
Speaker	25.9	2.0
State	0.7	21.5
Party	0.2	0.0
Justification	99.1	0.8
Label	0.0	0.0

All the samples from this dataset are from just one site and about one topic (namely politics), so this may make a model trained on this data not generalize well to other topics or styles. A few examples of the dataset are included below.

Description	**Example**
Statement	Says the Annies List political group supports...
Topic	abortion
Context	a mailer
Speaker	dwayne-bohac
State	Texas
Party	republican
Prior history	[0, 1, 0, 0, 0]
Justification	That's a premise that he fails to back up. Annie's List...
Label	false
Statement	Hillary Clinton agrees with John McCain...
Topic	foreign-policy
Context	Denver
Speaker	barack-obama
State	Illinois
Party	democrat
Prior history	[70, 71, 160, 163, 9]
Justification	Obama said he would have voted against the amendment...
Label	mostly-true

4.1 Data preprocessing

For our baseline model, the only preprocessing needed was extracting the textual claims and corresponding truthfulness labels from LIAR-PLUS, and mapping each word in each claim to its corresponding pretrained GloVe embedding. We downloaded the 50-dimensional GloVe embeddings that were trained on the Common Crawl and saved them to our Google Drive. The embeddings for a claim were averaged to create sentence embeddings. The truthfulness labels were converted into one-hot vectors to facilitate calculating model performance and accuracy.

For our first approach, additional preprocessing was needed. Any samples with missing data, such as justifications, were excluded. Textual claims, justifications, and prior history were extracted from the dataset. Words in the vocabulary were mapped to indexes, and indexes were mapped to pretrained GloVe embeddings. The truthfulness labels were converted in integers, from 0 to 5, so that they could be easily passed into the model.

Semantic features were extracted from the textual claim using Python libraries, such as PySentiStr (Thelwall et al.), and existing lexicons, such as EmoLex (Mohammad). These features included number of hedging words used, sentiment strength measured on a scale from 1 to 5 and either negative or positive, and the number of times words appeared that are associated with eight different emotions. We could not find a hedging lexicon, so the number of hedging words used was calculated with a hedging lexicon that we assembled ourselves. The features were assembled into into a matrix, and each vector was normalized. Extracting these features took a very long time, so we ran the whole process once, downloaded the data as txt files onto our Google Drive, and uploaded them from these txt files whenever we needed to use them after that.

The claims, justifications, metadata, feature vectors, and labels were assembled into a matrix, converted into Tensors, and divided into batches of size 64. After this, they were properly formatted to be used as input for the model.

For our second approach, the textual claims and justifications were tokenized and converted into Dataframe form.

5 Baselines

Our baseline model is a simple neural network that uses averaged GloVe sentence embeddings of the claims and classifies them on a 6-point truthfulness scale. The code associated with this model is contained in *BaselineModel.ipynb*. We chose this model over our initial idea of simply predicting the most common truthfulness label from the training idea because this model is a more useful comparison for complexity against our approaches. Our approaches employ neural networks, so it makes more sense for our baseline model to be a very simple neural network instead of based on a statistical measure.

The model has an input dimension of 50 (because each GloVe embeddings is 50 dimensions), one 150-dimension hidden layer, a softmax layer, and a 6-dimension output layer where each dimension represents a truthfulness label.

The same train, validation, and test sets of

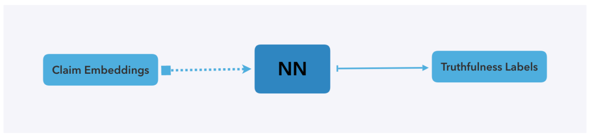


Figure 3: Baseline Model

LIAR-PLUS were used as those used by Alhindi et al. (2018). There are 10,240 samples in the train data, 1,284 samples in the validation data, and 1,267 samples in the test data.

Each claim is treated as a bag of words where each word is represented by a 50-dimensional GloVe embedding. The averaged sentence embeddings were converted into Tensors and given as input to the model. The 6-dimension output vectors were converted into one-hot vectors and compared with the one-hot vectors representing corresponding truthfulness labels to calculate accuracy and F1 scores. To calculate binary classification scores, the first three dimensions were all considered as ”false” and the last three were all considered as ”true”.

The loss function used is mean squared error, the learning rate is 0.001, and the number of epochs is 1000. We tuned the learning rate and number of epochs by trying incrementally smaller and larger numbers, and found that changes in these hyperparameters did not affect the results very much.

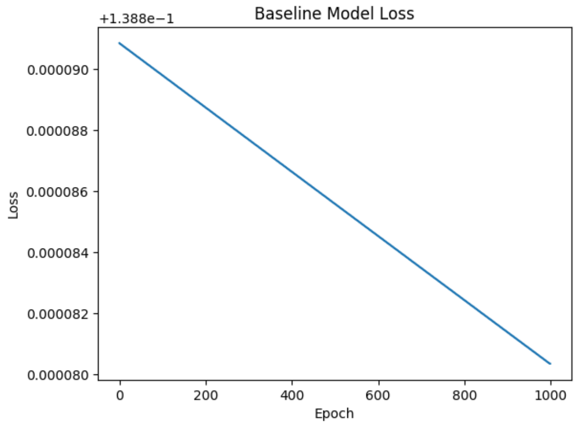


Figure 4: Baseline Model Loss

The accuracy results are comparable to the numbers one would expect for random guessing (50% and 17% for binary and 6-way, respectively). The model only predicts the labels ’false’ and ’mostly-true’ with accuracies of 4.5% and 14.3%, respectively.

	Binary	**6-way**
Accuracy	52.25%	19.18%

Table 1: Baseline model's accuracy

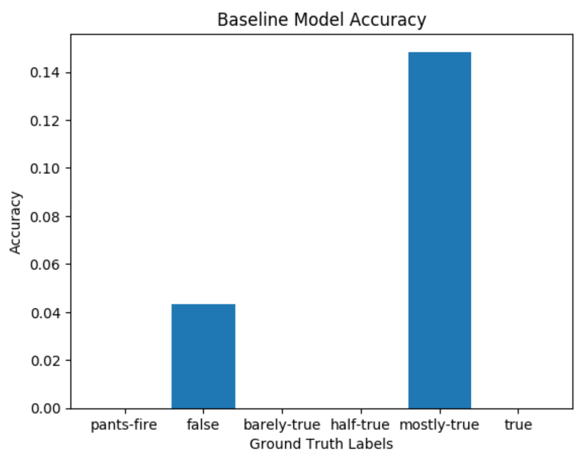


Figure 5: Baseline Model Accuracy

We have compared our F1 scores with the ones achieved by the BiLSTM in Alhindi et al. (2018) when only the textual claim was used. Our binary F1 score is surprisingly a bit higher by 0.4 points, while the 6-way score is lower by 0.4 points.

	Binary	**6-way**
Claim	0.60	0.23

Table 2: Alhindi et al. F1 scores

	Binary	**6-way**
Claim	0.64	0.19

Table 3: Baseline model F1 scores

6 Your approach

Our approach includes 2 different models, the major difference being that the first uses pretrained GloVe embeddings and the second uses contextualized embeddings such as BERT and ELMo. The same train, validation, and test sets were used as those used by the baseline model. We expect that it will fail similarly to the baseline model in that it will rely on prior data and predict labels that were more common in the training data, but we also expect that it will not rely on this as much as the baseline model and thus do better. We do expect it to fail in different ways, such as relying too heavily on metadata or feature vectors and not enough on the textual entailment between claim and justification.

The first model is a neural network architecture such as that in Alhindi et al. (2018) which we implemented ourselves. The associated code is contained in *RealModel.ipynb*. We had initially planned to implement a dual BiLSTM model where the claim's embeddings are given to one BiLSTM and the justification's embeddings are given to another, but we had trouble with this parallel architecture. The outputs of both BiLSTMs had to be concatenated with one another as well as with a feature vector before being passed to a softmax layer, and this additional level of complexity became an issue that was taking up a lot of time to try and figure out. Because of this, we decided to implement a model with a single BiLSTM, which Alhindi et al. (2018) showed had slightly lower performance than the dual architecture but did not perform significantly worse.

In this model, the embeddings of both the claims and justifications concatenated, padded to 200 dimensions, and given to a BiLSTM. The output from the BiLSTM is concatenated with a normalized feature vector, which is then passed to a softmax layer, and a 6-dimension probability vector is output. We based our implementation closely on that done by Alhindi et al. (2018), so we chose the same hyperparameters. It has 10 epochs, the loss function is cross-entropy loss, the optimizer is Adam, and the hidden layer is 32-dimensions. We used an existing BiLSTM implementation used for sentiment classification as a starting point for our model [1]. Our model uses NumPy, Pandas, PyTorch, and PyDrive, and we ran our experiments using Google Colab and its GPUs. There were runtime issues with Colab, but they were fixed by reducing batch size. We found loading the pretrained embeddings into the BiLSTM embedding layer to be more difficult than we had initially planned, as well as concatenating the BiLSTM output with a feature vector before passing it to a softmax layer, but both of these issues were solved. There was also a problem with the model not being in training mode when the loss.backward() function was called, but this seemed to be caused by the hidden layer not being detached in between batches and was promptly fixed.

[1]https://github.com/clairett/pytorch-sentiment-classification

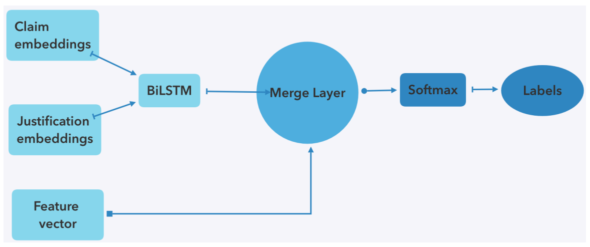


Figure 6: Our First Approach

Our implementation does work, although it has some interesting results. Regarding accuracy, the model did very well when metadata and feature vectors (M + F) were used. It scored about 20 points higher than the baseline model for both binary and 6-way classification. However, when only claim or justification embeddings (C + J) were used, the model performed very poorly and barely did better than the baseline model. When given this input, it scored 0 to 3 points higher for binary classification and 1 to 2 points higher for 6-way classification. This is barely better accuracy scores than could be expected for random guessing. Interestingly, this model performed better when justification embeddings were not given as input.

	Binary	**6-way**
C	55.10%	20.81%
C + J	52.63%.	21.13%
C + M + F	70.23%	42.27%
C + J + M + F	69.57%	39.97%

Table 4: Our first approach accuracy

Regarding F1 scores, we again compare our results to that of Alhindi et al. (2018). When metadata and feature vectors were used, our model performed as good as or slightly better. Our model scored 0 to 4 points better for binary classification and 8 to 18 points better for 6-way classification. However, when it was only claim and justification embeddings, our model scored 23 to 43 points lower for binary classification and 10 to 13 points lower for 6-way classification. This model indeed had much worse F1 scores than our baseline model when metadata and feature vectors were not used, with 27 to 47 points lower for binary classification and 1 to 6 points lower for 6-way classification.

	Binary	**6-way**
C	0.60	0.23
C + J	0.68	0.31
C + M + F	0.62	0.25
C + J + M + F	0.68	0.32

Table 5: Alhindi et al. F1 scores

	Binary	**6-way**
C	0.17	0.13
C + J	0.45	0.18
C + M + F	0.66	0.42
C + J + M + F	0.67	0.40

Table 6: Our first approach F1 scores

In our second model, we attempted to use contextual embeddings as a classifiers by using the AllenNLP library. Unlike the first model, this one does not use any metadata or features and instead only claim and justification embeddings. The code associated with the first iteration of this model is contained in *ModelWithAllennlpFromScratch.ipynb*. This followed the AllenNLP tutorial on the AllenNLP main page. Following the needed interface, we created a reader object and a task object. The reader transformed Pandas Dataframes into AllenNLP instances. The task object implemented the classifier. This version of the model was not successful, so the task and reader implementations were transferred to another notebook.

That notebook, *ModelWithContextualEmbeddings.ipynb*, used Homework 3[2] as a template and can use GloVe, ELMo, or BERT embeddings. On top of each embedding is a classifier layer that maps the embedding output to the dimension of the label space. For each model, a Dataframe was created that had the original samples and predictions. We did not implement binary classification or calculate F1 scores for this model because we did not have time to modify the template to get these statistics.

	6-way
GloVe	23.21%
ELMo	25.87%
BERT	24.98%

Table 7: Our second approach accuracy

[2]https://colab.research.google.com/drive/1nuaVF4CKMdPHEGrhSOharj$_X7MAs_VcS$

These results are also not much better than those of the baseline model. This model scored 3 to 5 points higher than our baseline model and 1 to 3 point higher than our first approach when it only used claims and justifications no matter which embeddings it used.

All of our models that used only claims or justifications (which includes the baseline model, the first approach with only claims and justifications, and the second approach) had similarly poor results which were only a few points above the numbers expected for random guessing.

7 Error analysis

Regarding our baseline model, because the only labels that it guesses are 'false' and 'mostly-true', it fails on any inputs that are not these. These two labels make up about 40% of the training data, so it seems to just be predicting two of the most common truthfulness labels. There doesn't seem to be any commonalities to the inputs that it fails on. The problem is probably mostly due to the fact that each classification is dependent on an averaged 50-dimensional embedding, which almost certainly does not have enough information to properly classify a statement as true or false.

Regarding our first approach, the fact that it performed so poorly when not given metadata and feature vectors and that accuracy is lower when it is given justification embeddings suggests that our model relied mostly on the metadata and feature vectors for the classification task. For example, C + J + M + F has an F1 score of 0.67 and 0.40 for binary and 6-way classification, respectively, while C + J scored much lower with 0.45 and 0.18. When justifications were not used, accuracy was generally higher by 1 to 3 points.

When the claim, justification, metadata, and feature vector were all used, there did not seem to be any commonalities between inputs that failed. For instance, sometimes the prior history metadata was very sparse, sometimes it was very dense, sometimes it was evenly distributed, and sometimes it wasn't. There didn't seem to be any semantic, syntactic, or length commonalities between claim and justification inputs either. When it did fail, the model predicted truthfulness scores that were too low (more false) instead of too high (more truthful). In the test data, 6.7% of the samples are 'pants-fire', 20.1% are 'false', 16.4% are 'barely-true', 21.3% are 'half-true', 19.2% are 'mostly-true', and 16.1% are 'true'. This model predicted that 7.1% are 'pants-fire', 26.4% are 'false', 11.8% are 'barely-true', 26.6% are 'half-true', 17.1% are 'mostly-true', and 11.0% are 'true'. Overall, it overestimates the number of 'false' and 'half-true' labels and underestimates all of the others. This is similar to the behavior of the baseline model, since 'half-true' and 'false' are the two most common labels in the training data and this model seems to be predicting these two labels 60% of the time. This shows how the model is influenced by prior data, but it overall is within 6 points of the correct percentage.

When only claim or justification were used, the model failed on many more inputs. It seemed more likely to be successful when there were numerical facts or specific named entities in the claim or justification. Again, in the test data 6.7% of the samples are 'pants-fire', 20.1% are 'false', 16.4% are 'barely-true', 21.3% are 'half-true', 19.2% are 'mostly-true', and 16.1% are 'true'. This model predicted that 1.6% are 'pants-fire', 28.8% are 'false', 12.2% are 'barely-true', 31.1% are 'half-true', 18.0% are 'mostly-true', and 8.3% are 'true'. With this input, the model is within 10 points of the correct percentage. This is similar to the behavior of the model when given claim, justification, metadata, and feature vectors, since it's overestimating the number of 'false' and 'half-true' labels and underestimating all the others, but to a more extreme degree. When the model only has claims and justifications, it relies even more heavily on prior data and less on the information encoded in the samples.

Description	**Example**
Statement	The number of illegal immigrants could be 3 million...
Prior history	[63, 114, 51, 37, 61]
Justification	Both figures are not within the range of possibility...
Label	pants-fire
C + J Label	true
C + J + M + F Label	barely-true
Statement	Says John McCain has done nothing to help the vets.
Prior history	[63, 114, 51, 37, 61]
Justification	While many veterans groups have had their...
Label	false
C + J Label	half-true
C + J + M + F Label	false
Statement	PolitiFact Texas says Congressman Edwards attacks...
Prior history	[2, 0, 0, 0, 0]
Justification	We informed Mackowiak of the date conflict. He...
Label	barely-true
C + J Label	barely-true
C + J + M + F Label	barely-true

Regarding our second approach, there also were not any obvious commonalities between inputs that failed. However, the models seemed to almost entirely predict the label 'half-true, which is the most common label in the training data and makes up 20.5% of it. When using GloVe embeddings, the model only predicts 'half-true' for every single claim. Because of this, it seems that the models are relying almost entirely on prior data and not using information stored in the embeddings to classify the claims. It seems unlikely that there isn't useful information in the embeddings, so it seems probable that there is a fault in the code so that the model is cannot access the embeddings or is ignoring them.

Description	**Example**
Statement	Sen. Bob Menendez voted to enact a new tax.
Justification	Santorum is correct when he says some...
Label	pants-fire
GloVe Label	half-true
ELMo Label	half-true
BERT Label	half-true
Statement	In the early 1980s, Sen. Edward Kennedy...
Justification	Former President Clinton said government..
Label	barely-true
GloVe Label	half-true
ELMo Label	half-true
BERT Label	barely-true

To summarize, all of our models that used only claims or justifications had very poor accuracy and F1 scores. They seemed to rely heavily on prior data seen in the training set and did not use the information stored in claim or justification embeddings. On the other hand, our model that also used metadata and feature vectors performed even better than the model presented in Alhindi et al. (2018). No matter what the complexity of the model, whether it was a simple one-layer neural network or a BiLSTM or BERT, the models all performed very similarly. This raises the question of if the claims and justifications are even necessary, given that the classification task could be performed so well with only metadata and semantic features. Conversely, our code could be flawed so that our models are unable to or are unwilling to access the information in the embeddings.

8 Contributions of group members

- Tessa: data collection and preprocessing, built and trained models, error analysis, wrote milestones
- William: data collection, built models, error analysis
- Zhixiang: data collection
- Nathan: data processing, created model schematics

9 Conclusion

Overall, this project was a very informative experience. We learned a lot about how to use Python libraries like PyTorch, NumPy, and PySentiStr as well as Google Colab, and how to troubleshoot associated issues. We also learned a lot about the current state of the automated fact-checking field and the variety of approaches that are used to tackle this classification task. While the homeworks were useful for learning about neural networks and how they work, this project really solidified our understanding of them. We built multiple neural network classifiers from scratch, which is much more work than simply filling out stub code, so by the end of the process we really understood our code and how it works.

Implementing a dual BiLSTM was more difficult than we expected, partly because there aren't as many online resources for this specific architecture and partly because we hadn't gone over any similar examples in class. It was much easier to implement a single BiLSTM because other implementations were easily available to get guidance from, and because we had gone over similar implementations of RNNs in class and in homeworks. Incorporating embeddings into our models was also surprisingly difficult, even when they were just pretrained GloVe embeddings.

We were surprised by our results because it seems like our models did not use any information from the embeddings representing the textual claim and justification, whether they were pretrained GloVe embeddings, ELMo, or BERT. It seems very unlikely that no useful information is held in the embeddings, as well as contradictory to the results of Alhindi et al. (2018), so our models' behavior is probably due to a fault in our code. It is then surprising that the metadata and feature vector held so much information that it could occasionally outperform the model of Alhindi et al. (2018).

If we could continue working on our project in the future, we would want to look more at what seems to be a problem with how the embeddings are used by the models. We would also like to try adding additional features, such as semantic similarity between claim and justification or the

Flesch-Kincaid readability score of a claim, and an additional attention layer. We would also want to try more hyperparameter tuning, as well as multiple hidden layers in the BiLSTM model to see how a deep neural network performs.

References

[1] Alhindi, T., Petridis, S., and Muresan, S. (2018). Where is your evidence: Improving fact-checking by justification modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 85–90.

[2] Bast, H., Buchhold, B., and Haussmann, E. (2017). Overview of the triple scoring task at the wsdm cup 2017.

[3] Dong, X. L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., and Zhang, W. (2015). Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949.

[4] Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., Hasan, S., Joseph, M., Kulkarni, A., Nayak, A. K., Sable, V., Li, C., and Tremayne, M. (2017). Claimbuster: The first-ever end-to-end fact-checking system. *PVLDB*, 10:1945–1948.

[Mohammad] Mohammad, S. Emolex. http://sentiment.nrc.ca/lexicons-for-research/.

[6] Nakashole, N. and Mitchell, T. M. (2014). Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1019, Baltimore, Maryland. Association for Computational Linguistics.

[7] Rashkin, H., Choi, E., Jang, J., Volkova, S., and Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. pages 2931–2937.

[Thelwall et al.] Thelwall, M., Buckley, K., Paltoglou, G., and Cai, D. Sentistrength. http://sentistrength.wlv.ac.uk/.

[9] Thorne, J. and Vlachos, A. (2017). An extensible framework for verification of numerical claims. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 37–40, Valencia, Spain. Association for Computational Linguistics.

[10] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

[11] Vlachos, A. and Riedel, S. (2015). Identification and verification of simple claims about statistical properties. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2596–2601, Lisbon, Portugal. Association for Computational Linguistics.

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

[13] Zhang, Y., Ives, Z., and Roth, D. (2019). Evidence-based trustworthiness. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 413–423.

[14] Zhou, J., Han, X., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2019). GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901, Florence, Italy. Association for Computational Linguistics.