Using Predictive AI to Detect Truthfulness in News Statements

Nicholas Shor
nshor@ucsd.edu

Dr. Ali Arsanjani
arsanjani@google.com

Abstract

The issue of fraudulent and misleading news has been an issue for as long as humans can remember, but today it is more rampant than ever with online news and social media giving everyone the chance to stay up to date with news worldwide. With the development of AI, there is the opportunity to either extenuate the increasing amount of misinformative news, or to mitigate it. Previous approaches have brushed the surface, with many fact-checking websites leading the way in debunking non-factual statements made by various sources, but the development of generative AI has opened new doors that have allowed for new techniques to develop. This project will continue to push the needle towards removing misinformation and disinformation from media sources through predictive AI and generative AI machine learning methods. By finding new data that is specific to this task, along with developing models addressing many different factors that go into detecting misinformation, this project will be a culmination of the most important factors contributing to mis/disinformation which is unlike many other projects previous. Thus far I have only been able to implement the predictive AI aspects of this project, but by the end of next quarter the result will be a generative AI interface that is able to intake a news article and return how false it is by providing text describing the falsities, along with a falsity score from 1-100.

Code: https://github.com/nshor47/DSC180A_Q1

1 Introduction

Throughout the internet, there are countless sources of news that people use every day to keep themselves updated on current events. The most prevalent are news websites and social media apps. These platforms have grown consistently as web applications have continued to develop, making them the primary news sources for people worldwide. The problem is that it has become easier than ever for misinformative posts and news to spread rapidly with the increase of news sources on the web. It is extremely difficult to detect wrong from right when the credibility of a piece of news, among other factors, cannot be easily detected. The goal of this project is to consider as many of the factors present and use these to paint a picture of where the article may be truthful versus misinformative. This will allow the user to take this information and make their best judgment on whether it is valid or not. Thus far, my process has led me to obtain much larger amounts of usable data through web scraping to train models on, which is an area that is still improving regarding this topic. My models developed so far have focused on using NLP methods that create embedding vectors of the text and title of an article to detect textual trends that may determine the level of misinformation. This misinformation level is scaled using PolitiFact's Truth-o-meter scale which has a range of classifications which are, True, Mostly-true, Half-true, Barely-true, False, and Pants-on-fire. This gives a much more accurate description of the truth value of an article in comparison to a simple binary classifier stating whether something is true or false. Thus far the performance of these predictive models isn't strong enough to validate deployment on their own, but the work of others and the combination of models will lead to a much more accurate final model that will help people determine the validity of a piece of news.

1.1 Literature Review

Previously there has been a large amount of research and work put into predictive tasks using data and well-known machine learning algorithms to try and detect misinformation. A major focus of my project so far has been on the Liar Liar dataset. A novel study was conducted by William Yang Wang in which he created this new benchmark dataset to be used for fake news detection. Wang (2017) drew information from PolitiFact.com to create a dataset with over 12,000 data entries of the statement, speaker, context, truth-label, and justification for some factual/non-factual statement (Wang 2017). The development of large datasets like this emphasizes how early on we still are in this process of quality fake news detection methods. This study also goes on and uses this dataset to try and classify similar statements into their correct truth-value category. Unfortunately, the best results Wang (2017) got were from Hybrid CNN models that used all the data present but still only got 27.4% accuracy (Wang 2017). Beyond this study, there have been adjustments to the Liar Liar dataset which have added context to the statements provided. The Liar Liar Plus dataset is an extension that has a justification added to the dataset for each entry, providing more context on the matter for any model developed to draw from. This dataset created by Alhindi, Petridis and Muresan (2018) took the Liar Liar dataset and added the

justification for the claim that was provided in the fact-checking article associated with the claim. This article explained why the statement some person made was true, false, or somewhere in between. To convert the textual data to usable features for prediction, the team ended up using unigram representations (Alhindi, Petridis and Muresan 2018). The deep learning model of choice was a Bi-directional Long Short-term Memory architecture that has previously proven to be successful with NLP-related tasks (Alhindi, Petridis and Muresan 2018). By using the extracted justification along with additional metadata, the team achieved 37% accuracy on the test set, a significant improvement from Wang's 27% accuracy. This is a huge increase and emphasizes the importance in providing additional context during model building.

1.2 Data

The data I am using is a combination of readily available data from previous studies, as well as data I was able to scrape and collect especially for usage on my own project. In the process of trying to maximize accuracy on the Liar Liar Plus dataset, I used a few datasets. The first was, of course, the Liar Liar Plus dataset (Alhindi, Petridis and Muresan 2018). This dataset had a statement made by an individual or post online, the associated truth label, subject, speaker, speaker's job, political party, location of the statement, and additional justification. It also had the counts of previous truth ratings of individuals. This was the baseline dataset I was working with (Figure 1).

	Json_File_ID	Truth_Label	Statement	Subject	Speaker	Speakers_Job	State	Party	Context_Venue_Location	Justification
0	2635.json	false	says the annies list political group supports ...	abortion	dwayne-bohac	State representative	Texas	republican	a mailer	That's a premise that he fails to back up. Ann...
1	10540.json	half-true	when did the decline of coal start it started ...	energy,history,job-accomplishments	scott-surovell	State delegate	Virginia	democrat	a floor speech.	"Surovell said the decline of coal ""started w...
2	324.json	mostly-true	hillary clinton agrees with john mccain by vot...	foreign-policy	barack-obama	President	Illinois	democrat	Denver	"Obama said he would have voted against the am...
3	1123.json	false	health care reform legislation is likely to ma...	health-care	blog-posting	NaN	NaN	none	a news release	"The release may have a point that Mikulskis c...
4	9028.json	half-true	the economic turnaround started at the end of ...	economy,jobs	charlie-crist	NaN	Florida	democrat	an interview on CNN	"Crist said that the economic ""turnaround sta...

Figure 1: LiarLiarPlus Dataframe

The next dataset I used was made from collecting news articles like those mentioned in the Liar Liar Plus statements through the Perigon (2023) News API. I was able to get

an API key for free with 500 API searches per month. I then used these 500 searches to create a mini dataset of the Liar Liar Plus with added context by finding more similar news articles (Figure 2). This was helpful for my context enrichment task. I also used a dataset that was scraped by my classmate, Jiang (2023), that contained all the truth ratings of all PolitiFact truthfulness speakers in a column called check_nums. The order of truthfulness was true, mostly true, half true, barely true, false, and pants on fire. This was very useful in my reputation factuality factor task (Figure 3).

	Truth_Label	Statement	content_content_1	content_author_1	content_source_1	content_summary_1	content_url
1046	false	says planned parenthood provides about 140 vis...	when it comes to pregnancy amid a pandemic the...	Dr. Ellen Stang	fredericksburg.com	While the pandemic has increased feelings of l...	https://fredericksburg.com/opinion/editorial/
10096	mostly-true	we bailed out johnson controls when we saved t...	when lord frost who recently resigned as brexi...	Andrew Neil	dailymail.co.uk	When Lord Frost, who recently resigned as Brex...	https://www.dailymail.co.uk/debate/article-103
7854	true	fourteen million americans mortgages are great...	albemarle county has committed more than 2 mil...		augustafreepress.com	\n\nThis program is facilitated through/by par...	https://augustafreepress.com/mortgage-relief-a

Figure 2: Perigon News API Data Distillation

	media	when/where	content	label	speaker	documented_time	percentages	check_nums	summaries	article
0	Instagram posts	stated on October 28, 2023 in a screenshot sha...	“Haaretz investigation reveals discrepancies i...	FALSE	Madison Czopek	October 31, 2023	['0%' '0%' '2%' '7%' '67%' '21%']	[5 3 16 54 473 152]	['Haaretz, an Israeli newspaper, said on X tha...	A viral Oct. 28 social media post claimed that...
1	Scott Walker	stated on May 30, 2023 in Interview:	“Wisconsin has historically ... and I think larg...	barely-true	Laura Schulte	October 31, 2023	['12%' '21%' '18%' '19%' '21%' '5%']	[26 45 39 41 44 11]	['Although Wisconsin has voted for more Democr...	In 2016, Wisconsin helped to swing the preside...
2	Instagram posts	stated on October 27, 2023 in a post:	“The airport in Salzburg, Austria, has a count...	FALSE	Ciara O'Rourke	October 30, 2023	['0%' '0%' '2%' '7%' '67%' '21%']	[5 3 16 54 473 152]	[]	A social media post poised to encourage people...
3	Viral image	stated on October 27, 2023 in an Instagram post:	Video shows Palestinians pretending to be corp...	FALSE	Ciara O'Rourke	October 30, 2023	['0%' '1%' '2%' '4%' '62%' '28%']	[4 13 35 53 745 336]	['This video is 10 years old and shows student...	The Gaza Health Ministry has said the Palestin...
4	Facebook posts	stated on September 25, 2023 in a Facebook post:	The life span of a wind tower generator lasts ...	FALSE	Loreben Tuquero	October 30, 2023	['0%' '1%' '4%' '9%' '59%' '23%']	[24 50 108 247 1519 594]	['A study by energy industry experts showed th...	Let's clear the air. Do wind turbine component...

Figure 3: Sean Jiangs Scraped PolitiFact Dataset

Outside of these datasets, I also did web-scraping of my own to collect data to feed into the Generative AI model I will be building next quarter. The data I collected was the text from 3000 news articles on PolitiFact, along with the date published, author, title, and a text summary. I also scraped my own PolitiFact truth-o-meter data that had the claimer, statement, truth value, summary of text, and text which is very similar to the Liar Liar dataset, but my scraping included the text and excluded a few other features.

2 Methods

2.1 Data Enrichment Techniques

A large part of improving performance on the Liar Liar dataset was by taking the dataset and extending it by creating new features based on what we already had to provide the model with enriched data to improve predictions. There are many different ways of doing this but I stuck to two methods.

2.1.1 Topic Modeling with LDA

The first technique I used was Latent Dirichlet Allocation (LDA). LDA is a generative probabilistic model that is commonly used for topic modeling. LDA assumes that each document in a corpus is a mixture of topics, and each topic is a mixture of words. I implemented LDA in my Liar Liar Plus testing by using it to uncover latent topics in the statements provided by speakers, as well as to identify latent topics in the additional news articles I found to enrich context.

This implementation required the preprocessing and tokenization of the text before converting the documents into a bag-of-words representation. Pre-processing was simply converting text to lowercase and removing punctuation. The model parameters were fine-tuned during the training process to optimize performance (Figure 4). The dominant topics for each piece of text were identified by calculating the probability that the topic matched the text and selecting the topic with the highest probability. The dominant topic number and highest probability were then added as new features to the dataset (Figure 5).

```
#LDA
documents = liar_plus_train['Statement'].apply(preprocess_text)
texts = [document.split() for document in documents]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
lda_model_body = models.LdaModel(corpus, num_topics=30, id2word=dictionary, passes=15)

topic_distributions = [lda_model_body[doc] for doc in corpus]
dominant_topics = []

for topic_dist in topic_distributions:
# Sort topics by probability and get the most likely topic
    dominant_topic = max(topic_dist, key=lambda item: item[1])
    dominant_topics.append(dominant_topic)

# Extract the topic number and probability from the dominant topic
topic_numbers = [topic[0] for topic in dominant_topics]
topic_probabilities = [topic[1] for topic in dominant_topics]

# Add the topic information as new columns
liar_plus_train['DominantTopicBody'] = topic_numbers
liar_plus_train['TopicProbabilityBody'] = topic_probabilities
```

Figure 4: LDA Code Snippet

	DominantTopicBody	TopicProbabilityBody	Statement
0	10	0.414420	says the annies list political group supports ...
1	14	0.151673	when did the decline of coal start it started ...
2	10	0.283193	hillary clinton agrees with john mccain by vot...

Figure 5: LDA as features

Using LDA helped to extract latent topics in the dataset which provided the model more context. This will be especially useful when we implement the Generative AI aspect of the project, as the associated LLM used will have knowledge of the specific topic and can draw from it to make its predictions.

2.1.2 Contextual Enrichment

The second data enrichment technique I used was Contextual Enrichment. Put simply, Contextual Enrichment is just adding external information related to entities being observed in the dataset. In this case, the additional context I provided was the text of news articles on the same topic. The process of obtaining these news articles began with a keyword extraction model called KeyBERT. This gives back the most important and relevant words in a given text. I used this to extract the top three keywords and then used them to query the Perigon (2023) News API to find articles on the same topic (Figure 6).

The obtained text of the news articles was then added as new columns in our dataset and additional relevant information such as the article's URL, author, summary, and source were also kept for potential further analysis. At most three news articles were obtained for each data entry.

This implementation's goal was to add additional text from different sources and backgrounds to give the model more context on the topic at hand when trying to make predictions. The resulting dataset should provide a more holistic and well-rounded view on the topic which should allow for a less biased prediction to be made.

```python
kw_model = KeyBERT()
for index, row in equal_random_sample.iterrows():
    statement = row['Statement']

    keywords = kw_model.extract_keywords(statement, top_n=3)
    doc_keywords = ' '.join([keyword for keyword, score in keywords if score > 0.01])

    tokenized_words = word_tokenize(doc_keywords)
    keyword_1 = tokenized_words[0]
    keyword_2 = tokenized_words[1]
    keyword_3 = tokenized_words[2]

    API_KEY = "89a2692d-ba03-4aee-9a16-ea725f927467"
    url = f"https://api.goperigon.com/v1/headlines?q={keyword_1} AND {keyword_2} AND {keyword_3}&from=2022-01-01&apiKey={API_KEY}"

    resp = requests.get(url)
    cluster_count = 0

    for cluster in resp.json().get('clusters', []):
        if cluster_count >= 3:
            break  # Stop iterating if we've reached the limit
        cluster_count += 1

        hit_count = 0
        for hit in cluster.get('hits', []):
            content_url = hit.get('url', '')
            content_content = hit.get('content', 'content')
            content_author = hit.get('authorsByline', '')
            content_source = hit.get('source', {}).get('domain', '')
            content_summary = hit.get('summary', '')
```

Figure 6: keyBERT and Perigon API implementation

2.2 Prediction Models

2.2.1 N-grams

The first model implemented in this project was a straightforward n-gram text embedding model. This approach allowed for the transformation of textual data into vector representations, enabling the model to make predictions. To prepare the textual data for embedding, a preprocessing step was applied to convert the text to lowercase and remove punctuation. Following this, n-grams, specifically bi-grams and quad-grams, were extracted from the preprocessed statements. These n-grams were used to train a Word2Vec model with a vector size of 100, a window of 6, and a minimum word count of 2 (Figure 7).

```python
#creating ngram model for statement
liar_plus_train['Statement'] = liar_plus_train['Statement'].apply(preprocess_text)
headline_split = [text.split() for text in liar_plus_train['Statement']]
model_headline = Word2Vec(headline_split, vector_size=100, window=6, min_count=2)
model_headline.save("word2vec.model_headline")
liar_plus_train['bi-gram-embeddings-statement'] = liar_plus_train['Statement'].apply(lambda x: get_ngram_embeddings(x, 2, model_headline))
liar_plus_train['quad-gram-embeddings-statement'] = liar_plus_train['Statement'].apply(lambda x: get_ngram_embeddings(x, 4, model_headline))
```

Figure 7: N-grams code snippet

The generated n-gram embeddings were incorporated into the dataset as additional features, providing richer contextual information. The embeddings were then flattened and individual n-gram values were isolated into separate columns. For each n-gram type (bi-gram and quad-gram), a set of columns was created to represent the individual values within the embeddings (Figure 8). This allowed for a detailed examination of the impact

of specific n-grams on the model's predictions.

Statement	bi-gram-value-1	bi-gram-value-2	bi-gram-value-3	bi-gram-value-4	bi-gram-value-5	bi-gram-value-6	bi-gram-value-7	...
says the annies list political group supports ...	0.205283	0.761333	0.595609	-0.036008	0.201812	-1.104238	0.995440	...
when did the decline of coal start it started ...	-0.318123	0.980255	0.604931	0.269575	0.231531	-1.129731	0.539515	...

Figure 8: N-grams feature representation

The final dataset, augmented with n-gram embeddings, was divided into training and testing sets. The Random Forest classifier was selected as the predictive model, with the optimal classifier chosen based on its accuracy on the test set. This n-gram text embedding model served as a foundational approach to capturing the patterns and structures within the textual data we had.

2.2.2 Reputation

To obtain the predictions for the truthfulness of a statement based on the speaker's previous reputation I took two approaches. The first was by simply obtaining the most common truthfulness rating for that specific speaker. To achieve this, the ratings associated with each speaker's statements were extracted from the "check_nums" column, representing counts for different truth categories: TRUE, mostly-true, half-true, barely-true, FALSE, and pants-fire. The label for each speaker was then determined based on the maximum count across different truth categories. This label represents the overall, most likely classification of the speaker's statements.

The second method followed the same process, except all speaker's previous truthfulness ratings were taken into account and weighted (Figure 9). The order of weighting is the same as the check_nums column ratings: TRUE, mostly-true, half-true, barely-true, FALSE, and pants-fire. The count of each label is then multiplied by the weighting scale and then divided by the total number of ratings for that speaker. The final label is then determined by the rating scale shown in Listing 1 below. These methods provide us with a much simpler prediction algorithm that is still very effective for this given task.

Listing 1: Label Prediction Code

```
# Ratings Extraction
def predict_label(confidence):
    if confidence < 0.166:
        return "pants-fire"
    elif confidence < 0.33:
        return "FALSE"
    elif confidence < 0.5:
        return "barely-true"
    elif confidence < 0.666:
        return "half-true"
    elif confidence < 0.833:
        return "mostly-true"
    else:
        return "TRUE"
```

```
def calculate_confidence(row):
    weights = [1, 0.8, 0.6, 0.4, 0.2, 0]
    total_ratings = row['TRUE_counts'] + row['mostly-true_counts'] + row['half-true_counts'] + row['barely-true_counts']
    + row['FALSE_counts'] + row['pants-fire_counts']

    try:
        confidence = sum(row[col] * weight for col, weight in zip(['TRUE_counts', 'mostly-true_counts', 'half-true_counts',
                                                                   'barely-true_counts', 'FALSE_counts', 'pants-fire_counts'], weights))
        confidence /= total_ratings
        return predict_label(confidence)
    except (ValueError, TypeError, ZeroDivisionError):
        return 'error'
```

Figure 9: Reputation weighting code snippet

3 Results

3.1 Baseline Model (LDA and n-grams)

First, we will look at the performance of the model using LDA and n-grams on just the statement provided in the LiarLiarPlus dataset. This was made to be my baseline model. After using the RandomForestClassifier we see in Table 1 below that I was able to get a 23% precision, 23% recall, and 22% F1-score. Additionally, we can see the individual class precision, recall, and F1-scores.

3.2 Contextual Enrichment Model with LDA and n-grams

Next, we have our similarly processed model using LDA and n-grams, but this time we are including the additional news data we collected from the Perigon (2023) News API. I as only able to get up to 3 news sources for 500 rows of data from the LiarLiarPlus dataset, so the training and test data is far less robust. Using another RandomForestClassifier, as seen in Table 2, I was able to achieve 21% accuracy, 21% recall, and 22% F1-score.

Table 1: LDA and N-grams Baseline Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Pants-fire	0.11	0.03	0.04	155
False	0.24	0.27	0.26	393
Barely-true	0.22	0.25	0.23	293
Half-true	0.24	0.27	0.25	446
Mostly-true	0.24	0.31	0.27	378
True	0.23	0.13	0.16	371
Accuracy			0.23	2036
Macro avg	0.21	0.21	0.20	2036
Weighted avg	0.22	0.23	0.22	2036

Table 2: Contextual Enrichment Model Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Pants-fire	0.25	0.27	0.26	11
False	0.00	0.00	0.00	8
Barely-true	0.5	0.27	0.35	15
Half-true	0.11	0.22	0.14	9
Mostly-true	0.18	0.17	0.17	12
True	0.3	0.27	0.29	11
Accuracy			0.21	66
Macro avg	0.22	0.20	0.20	66
Weighted avg	0.25	0.21	0.22	66

3.3 Truthfulness Reputation Model

Lastly, we have our truthfulness reputation statistical models. These required no training as they are just aggregating scores of speaker's previous truthfulness ratings. The following tables show that the maximum count labels model achieved 51% accuracy, 51% recall, and 50% F1-score (Table 3). The weighted model on the other hand achieved 44% accuracy, 44% recall, and 45% F1-score (Table 4).

Table 3: Maximum Reputation Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Pants-fire	0.65	0.32	0.43	3237
FALSE	0.57	0.68	0.62	7285
Barely-true	0.49	0.36	0.42	3961
Half-true	0.46	0.43	0.45	4124
Mostly-true	0.42	0.59	0.49	3772
TRUE	0.50	0.49	0.49	2911
Accuracy			0.51	25290
Macro avg	0.52	0.48	0.48	25290
Weighted avg	0.52	0.51	0.50	25290

Table 4: Weighted Reputation Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Pants-fire	0.66	0.33	0.44	3237
FALSE	0.63	0.56	0.60	7285
Barely-true	0.32	0.37	0.34	3961
Half-true	0.29	0.60	0.39	4124
Mostly-true	0.44	0.36	0.39	3772
TRUE	0.83	0.23	0.37	2911
Accuracy			0.44	25290
Macro avg	0.53	0.41	0.42	25290
Weighted avg	0.52	0.44	0.45	25290

4 Discussion

Overall, the performances of my LDA and n-gram models were quite disappointing. A score of 23% and 21% respectively is only a small improvement from a model that predicts one truth value for the entire dataset. The initial baseline model is a decent performer due to the lack of additional features used. In fact, in Wang's paper about the Liar Liar Dataset he

was only able to achieve a 27% accuracy score using all the features present (Wang 2017). This makes my baseline model seem quite good, but an accuracy score of 23% is still far too low to validate deployment. Additionally, in the Liar Liar Plus paper, it was mentioned that uni-grams were the embedding method of choice for their experiment (Alhindi, Petridis and Muresan 2018). It is possible that this decision would have improved my model's accuracy as well, but it didn't seem logically like the right decision in my opinion. The puzzling thing about my secondary LDA and n-gram model is that its performance decreased when I added more context. This is a bit confusing, but it could be because of a few reasons. It is possible that the extreme decrease in dataset size made the model far worse at making predictions, which is very likely. With just a small subset of data to train and test on, we are unable to get very meaningful results. Another potential drawback could have to do with the Perigon (2023) News API search process. Since the Liar Liar Plus dataset did not contain the dates of the statements made, I was unable to search for news articles that came out around the same time as the statement. I had to instead just search for articles in the past 2 years or so. This means that while the additional news articles added context, that context may be highly skewed since the article may have come out far before, or after the statement was said. This means our contextual enrichment may be damaging as it is providing the wrong context for the model to draw from. This leads to a final potential major drawback, and that is model choice. With the development of newer and more powerful models in recent years, the choice of using n-grams may be a bit outdated for this specific task. In future experiments, I would incorporate the use of transformers, such as a pre-trained BERT model for example. These models are far more effective at sifting through noisy features and finding signal that may be useful for prediction tasks such as this.

On the other hand, the performance of my model leveraging the previous reputation of a speaker's statements was fantastic. Although the model is extremely basic and simplistic in nature it was able to significantly outperform both my other more complex models. This makes sense logically, as if you tell 100 lies for every truth it is far more likely that your next statement will be a lie. This was the thought process behind this model. The problem with taking the most frequent truthfulness value for a speaker though is that it leaves little room for change in an individual which is something quite common in the world. If a person goes from being honest in all their statements, then suddenly begins to spread lies like wildfire then this model would perform horribly. This is undeniably possible in the world, which is what brought me to the weighting of the previous truthfulness values of an individual. This method is far more all-encompassing of the data available. It provides a weight from 0-1 for all the different truth values of previous statements and combines it to a final score which determines a future statement truthfulness rating. This undeniably has drawbacks as well, such as the fact that coming across data like this may be extremely difficult for many individuals. Additionally, if a user has no statements rated previously then neither of these models can be used as they will have nothing to draw their predictions from. Though overall this simplistic method far outperformed that of numerous others out there trying to achieve the same result, it still has much room for improvement.

5 Conclusion

In conclusion, this project has paved the way for future projects and developments of the same type. The pipeline of contextual enrichment and LDA is undoubtedly useful for enriching future datasets, which is necessary for the increase in accuracy for tasks such as this. These methods also provide far more information for models to draw from which is required to drastically improve performance. This project also showed many of the drawbacks of certain implementations which is invaluable information for me and others.

The entire process has also opened my mind to a plethora of new opportunities for improvement. From improved model selection and tuning, to additional data collection, to more in-depth factuality factor exploration, I now have the tools and ability to improve in all these areas as I continue my project in the following quarter. Additionally, the ensembling of methods from my peers working on the same project will drastically improve performance as we can leverage each other's models and ideas to build a final model that takes all factors into account. Finally, the addition of the generative AI piece is something I believe is critical in tackling the issue of fake news in the future. All projects in this field should begin to leverage this new technology as it is more powerful than ever and could be a major leap forward toward dismantling fake news in our society.

References

Alhindi, Tariq, Savvas Petridis, and Smaranda Muresan. 2018. "Where is your evidence: Improving fact-checking by justification modeling." In *Proceedings of the first workshop on fact extraction and verification (FEVER)*.

Jiang, Zhixing. 2023. "politifact data combined."

Perigon. 2023. "Perigon News API." https://www.goperigon.com/

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