Structured KO Project

Alan Zheng, Desirae Serrano-Diaz
Professor Pu
CS 4365 Summer 2024

Say No to Fake Info

Table of Contents

Introduction **3**
Motivation **5**
Related Work **7**
System Architecture **9**
Initial Snorkel Pipeline 11
Labeling 11
Update Snorkel Pipeline 11
Manual Verification Process 12
Train New Expert Models 12
Assemble 12
Compare the Performance 13
Datasets **13**
NELA Dataset 13
FNC Training and Test Datasets 14
JSON Datasets 17
Method **18**
Provided Dataset 19
Annotation and Labeling 19
Weak Supervision and Snorkel 21
Model Training 22
Evaluation 23
Schedule Breakdown **24**
Results **25**
Future Research **27**
Skills Learned **28**
References **28**

Introduction

The Knowledge Obsolescence project, also known as KO project, addresses the significant challenges faced when trying to detect and counter misinformation. The scope of this project is specifically in the context of the COVID-19 pandemic. The rapid spread of misinformation poses a threat to the public and public safety since it may undermine the efforts to track, manage, and handle outbreaks effectively. The KO project artificial intelligence to combat, counteract, and mitigate the effects of COVID-19 misinformation.

Machine learning is at the core of the KO project. Traditional machine learning models have long been used for misinformation detection, but they are usually built on static datasets, which in the age of information and social media, would lack the adaptability needed to handle a constant stream of new data. The inability for static models to process unstructured data, which is generated in real time, is a limitation of the traditional machine learning model as it can significantly impact the effectiveness of the model when trying to keep pace with the evolving misinformation landscape. Another limitation of typical machine learning models is its nature to slowly adapt. With so much incoming information, the model can become outdated and inaccurate very quickly. Additionally, the models are extremely resource intensive which can create a need for large, hand labeled datasets. This makes the traditional model much less scalable since training would become very tedious.

These limitations can carry over to the knowledge obsolescence project so there must be a more flexible and adaptive approach when handling COVID-19 information. Thus, we employ weak supervision techniques, which allow for integrating real time data and continuous refinement of the model. This helps to ensure that the detection system can evolve along with the data and information being presented in this volatile landscape.

The KO project aims to solve these limitations by using a dynamic and adaptable solution. Through the use of weak supervision with Snorkel, we are able to allow the model to evolve with data. Snorkel allows the project to generate labeled training data on its own by using several imperfect heuristics, known as a labeling function. This technique helps to circumvent the need for large manually labeled datasets and allows for a continuous refinement of detection models.

The methods section of the report will further explore the techniques used in the project. The idea of the dynamic approach is to first identify and collect data for a dataset. In the case of the KO project, we used Twitter data from the early months of COVID-19 which were preprocessed for the model. Next was to annotate and label the data which was used to establish a "ground truth" benchmark. Then we use weak supervision with Snorkel to generate some more labeled datasets. After new labels have been made, new models are trained using the labeled dataset and are continuously refined based on manual verification and feedback. Finally, the effectiveness of the model is evaluated through a series of performance metrics. This roughly summarizes the innovative technique of the dynamic approach for misinformation detection.

There are many advantages to using this approach as opposed to the traditional machine learning model approach. It allows for real time adaptability so the model can detect misinformation patterns even if its given new information. It also is much more scalable as weak supervision does not require extensive manual labeling, making it easier to yield labeled data. Finally, there is an improved accuracy since there is continuous refinement.

The knowledge obsolescence project represents a significant advancement in misinformation detection, setting a new standard for dynamic and adaptable solutions in informatics. Our project combines machine models that are both innovative and advanced. And

as a result, making our project a great resource to fight the spread of misinformation. The following sections of this paper will outline our experiences and workflow our structured KO project.

Motivation

It is currently the year 2024, and in this time, the world is at an all time technology high. Technology today is unlike anything that has been seen before from self driving cars to using face detection as a key. One technology that is evolving everyday from usage increases to features is social media, an environment that is completely online and where information is shared. Merriam-Webster defines social media as "forms of electronic communication through which users create online communities to share information, ideas, personal messages, and other content .

The first social media application was called Six Degrees, and was the first social media that made its appearance to the world in 1997. Six Degrees was a platform where users could personalize their profiles and meet others. At the time of Six Degrees release, only 5 percent of American adults were using social media. Now in 2024, that number has exponentially increased to 84% among American adults.

In 2024, trends have shown that the usage of social media has grown and will continue to do so. Today, over half the word engages in online networking through different forms of social media. The University of Maine in 2023 conducted a survey that revealed 59.9% of global users are online. The large growth of social media participation raises concern over the rising statistics

of fake information being spread. Fake information, also known as fake news, is false or misleading information that is presented as accurate or real news. Social media was initially created with the purpose of allowing people to connect, since then it is now an environment that users can use as a great resource to stay updated or learn about new information as it emerges. But with this comes the con that false information is able to spread very easily and is difficult to fight.

The problem of fake news becomes more dangerous during threatening events such as global health crises. It is important that everyone has access to reliable and accurate information in scary times. The COVID-19 pandemic showcased the consequences of what occurs when misinformation is spread. False claims had spread like wildfire on social media with examples being that COVID-19 was being caused by 5G networks, anti vaccine misinformation rumors emerged about infertility or microchips, and dangerous “cures” like drinking water mixed with cocaine were being supported. False news items such as these spread rapidly, making it more complicated for the world to overcome the pandemic.

Our team recognized these challenges,which led us to approach our project with a strong sense of urgency to address the spread of misinformation. Our motivation is rooted in making sure the truth remains evident and accessible. The COVID-19 pandemic exposed a flaw in the techniques that were being used by the world to verify information, noting that they are slow to adapt.

Our team took a proactive approach during our Knowledge Obsolescence project with the purpose being to combat misinformation by leveraging up to date weak supervision techniques.

Our goal is to make sure that our contributions and project results assist in making a stride in the field of information detection.

Our Knowledge Obsolescence project focuses on using techniques to keep our models up-to-date in accurately identifying false information from real news. By using machine learning techniques to continuously update and retrain fake news detection models on newly labeled data, we will solve the problem of knowledge obsolescence in fake news detection models, allowing them to stay up-to-date and accurate as new information and events emerge over time. With the benefits being:

Continues Learning: It allows fake news detection models to stay up-to-date by continuously learning from new data as it emerges.

Efficient Labeling: Manual labeling of large datasets is avoided by using techniques that can automatically label some of the new data.

Scalable and Cost-effective: This automated labeling approach makes it feasible to frequently retrain and update the models in a scalable, cost-effective manner.

Relevant Work

Before we deep dive into the specifics of our project, our team wants to highlight similar work and studies done by other researchers that combat fake news using detection techniques addressing Knowledge Obsolescence.

The New-Normal of Fake News: Enhanced and Sustainable Multi-Expert Detection through Timely Adaptation to Counter Knowledge Obsolescence

In their paper, authors Abhijit Suprem and Calton Pu introduce a new, complex system called Argus that was designed to combat the challenge of Knowledge Obsolescence in fake news detection. The Argus system uses Lipschitz's smoothness properties to scan data and detect shifts and divergence, as they are strong indicators of Knowledge Obsolescence. Argus has the ability to adapt to new information and evolving misinformation patterns though maintaining a dynamic team of submodels that are continuously generated and updated.

To prove the efficiency of Argus, the authors conducted an experiment using large fake news datasets, including one of COVID-19 and 11 others. The results found that Argus significantly outperformed existing models as Argus has an improvement 2x more effective than static classifiers and 1.3x more effective than fixed window classifiers. Due to Argus's ability to adapt continuously, it is a great solution to combat misinformation as it will be able to keep up with the dynamic landscape. Argus is a great example that in order to continuously fight fake news, innovative approaches are required in order to keep current with changes.

Reinforced Adaptive Knowledge for Multimodal Fake News Detection

This study by Abhijit Suprime and colleagues examine the effectiveness of using pre-trained and fine-tuned across various datasets. The authors of this paper found that these models struggle with the challenges of making accurate predictions on unseen data, bringing to light the issue of Knowledge Obsolescence (Zhang et al 2024). A method called KMeans-Proxy was introduced as a solution to address Knowledge Obsolescence. KMeans-Proxy uses K-means

clustering to identify overlapping subsets of unseen data that can be reliably classified. This makes it easier to manage KO by determining which data points require additional techniques to determine truthfulness such as active learning, weak supervision, or manual annotation. KMeans-Prox identifies and adapts to new data patterns and it is because of this the method enhances the models ability to remain relevant and accurate over time.

Application of Artificial Intelligence Techniques to Detect Fake News: A Review

In this study, the authors Md Rafiqul Islam, Shaowu Liu, Xianxhi Wang, and Guandong Xu provide a comprehensive overview of multiple deep learning techniques that are used to combat misinformation through accurate detection (Berrondo-Otermin). The paper detailedly discusses the evolution of misinformation detection methods and the importance of models being able to adapt to new and emerging patterns of misinformation, as misinformation is continuously evolving. In order to stay up to date and relevant, all detection models need to undergo continuous updates to keep pace with how misinformation is changing. It is important for fake news detection systems and techniques to stay up to date with misinformation because it is forever & continuously evolving. Within a short time frame, static models can become out of date and not know how to handle unrecognized forms of misinformation. A few machine learning techniques include deep learning, NLP, ensemble learning, transfer learning, and graph-based approaches.

System Framework

The system framework integrates several machine learning models and weak supervision techniques to first identify and then classify fake news content efficiently as well as adaptively.

The system takes advantage of Snorkel's ability to generate labeled datasets through the provided imperfect heuristics, called labeling functions. These labeling functions range from regular pattern matching all the way to advanced machine learning models and they can capture aspects of new content that traditional machine learning models can't. The Snorkel pipeline, which was provided by the TA's, includes pretrained models with the NELA dataset, which was the initial labeling function. We then further developed this dataset to enhance the effectiveness of the model. After integrating these into Snorkel, a set of probabilistic labels was generated which reflected the different perspectives and heuristics. Then, we used a manual verification process which ensured that the machine generated labels align well with our human judgment. Afterwards, a new, expert, model can be generated. The final pipeline integrates these models all together, which results in a system that combines the heuristics, labeling functions, and expert models. This comprehensive system is specifically designed for iterative improvement and its effectiveness is constantly being measured. The overall flow of the system can be described by the following diagram.

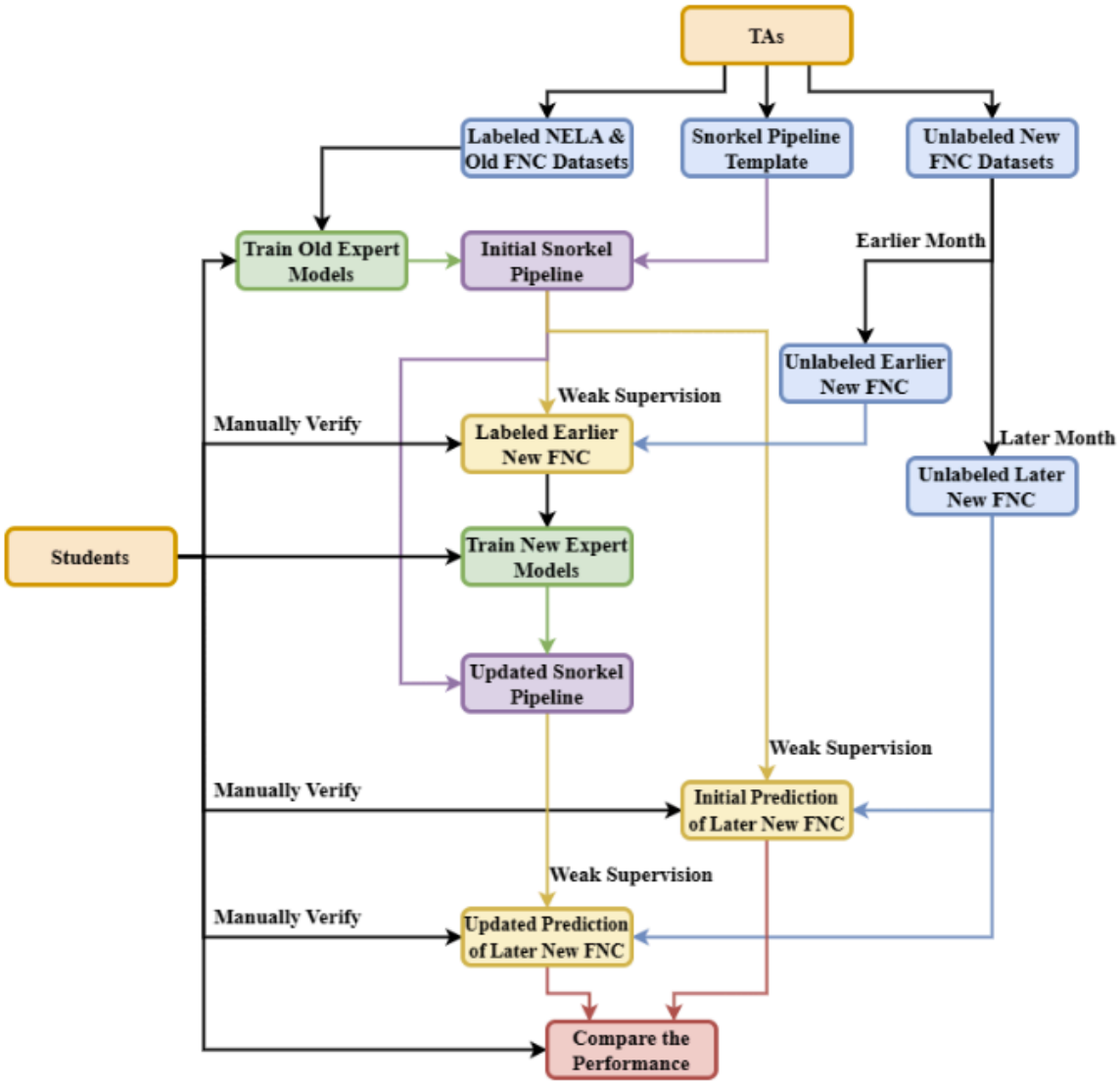

Figure 1: Knowledge Obsolescence Workflow

Initial Snorkel Pipeline

The initial Snorkel pipeline is the starting component of the structured KO project. It includes pretrained models based on the NELA dataset, which is a set of Twitter posts about COVID-19 or that were posted during the initial COVID-19 months. The pretrained model serves as the initial labeling function for Snorkel. This enables Snorkel to generate labeled data

on its own from new, unlabeled data. The base pipeline takes in, preprocesses, and labels data which sets up for later stages to work on. This setup creates robustness, helping the misinformation detection by weak supervision techniques.

Labeling

The labeling part of the KO project is focused on creating additional labeling functions which can help to enhance the initial Snorkel base pipeline. These labeling functions can vary by complexity, but are designed to capture aspects of new content. This can include textual patterns like emojis or caps lock and source reliability indicators like news outlets or CDC. This process uses advanced natural language processing techniques to identify these patterns. Ultimately, this component also helps to improve the accuracy and dynamic nature of misinformation detection.

Update Snorkel Pipeline

The updated Snorkel pipeline phase incorporates the newly developed labeling functions into the Snorkel pipeline. This stage uses Snorkel's robust abilities to combine outputs of individual labeling functions. By considering a series of factors, Snorkel generates probabilistic labels for data. This process ensures the pipeline can effectively identify and find deeper insights in the patterns found within the dataset like tone.

Manual Verification Process

The manual verification process is crucial in the KO project. This is the phase where we annotate a subset of data to establish a "ground truth" benchmark. This subset acts as a reference point for evaluating the effectiveness of Snorkel's labels. Snorkel can then iteratively improve the labeling functions based on these metrics by comparing machine generated labels against human

judgment. We can find optimal labeling functions that can be used to label data much faster and efficiently than if we were to manually label every datapoint. Overall, this enhances the quality and precision of misinformation detection.

Train New Expert Models

The training of new expert models in the KO project is when the enhanced labeled dataset from the Snorkel pipeline is used to train a new, expert, model. The training process involves using advanced techniques in either BERT or ALBERT. In our case, we used the ALBERT function to train the new expert model. This refinement ensures that the models are finely tuned to handle misinformation correctly. This process also allows for training and retraining for dynamic and adaptive purposes.

Assemble

The assemble part of the KO project brings together all the components of the system into one framework. This integrates all the components previously outlined into a dynamic and adaptive misinformation detection model. This creates an effective and scalable solution for combating misinformation.

Compare the Performance

The compare performance stage is the part of the KO project where the strength and effectiveness of the model is compared. This step allows for a comparison between predictions made by the initial pipeline to the ones made in the updated pipeline. The metrics include precision, recall, and F1 scores which are used to assess improvements in the model. This

rigorous evaluation helps to make sure that the pipeline is as effective as possible in identifying misinformation.

Datasets

In order to complete our project, our group utilized many different datasets that are a perfect example of real word information- specifically information presented during the COVID-19 pandemic in the form of tweets from twitter. Tweets were a great source of information to use for this project because each tweet is a source of real time data with opinions and sentiment. This allowed us to gain stronger insights into the spread of misinformation.

The datasets used are extremely/notably diverse, differing in content and structure. This enhances the depth and breadth of our analysis.

NELA Dataset

For our Knowledge obsolescence project, our group utilized two different NELA Datasets. NELA is a News Landscape dataset that is substantial in content. The two NELA datasets we used were one with content from November 2020 and one from December 2020. The content itself from the two data sets varied tremendously even if they were one month span from each other, proof that misinformation is evolving, however the content type was the same in both. Both NELA datasets were labeled text from multiple new's source outlet's, consisting of a 0 for false and 1 for real, making the NELA datasets have great value in our project. These NELA datasets had a large role in our project as they were the primary training resource for our pre-trained models. By using this dataset to train, our models were able to better analyze what constitutes as unreliable and reliable news.

FNC Training and Test Datasets

For our Knowledge Obsolescence project we used four separate FNC datasets with FNC standing for Fake News Covid. These four datasets varied by month and year, including two new datasets (January 2022 and February 2022) and two older ones (April 2021, and May 2021).

The old FNC datasets were labeled and contained types of information such as text and label. The New FNC datasets were similar in the types of information they contained, such as tweet text , language used, retweet count, user information, reply status, and tweet id. However, they differed greatly within the specific inputs within these fields.

Despite the data being noisy and very repetitive, our team was able to clean and preprocess both of the New FNC datasets to make them valuable for our project.

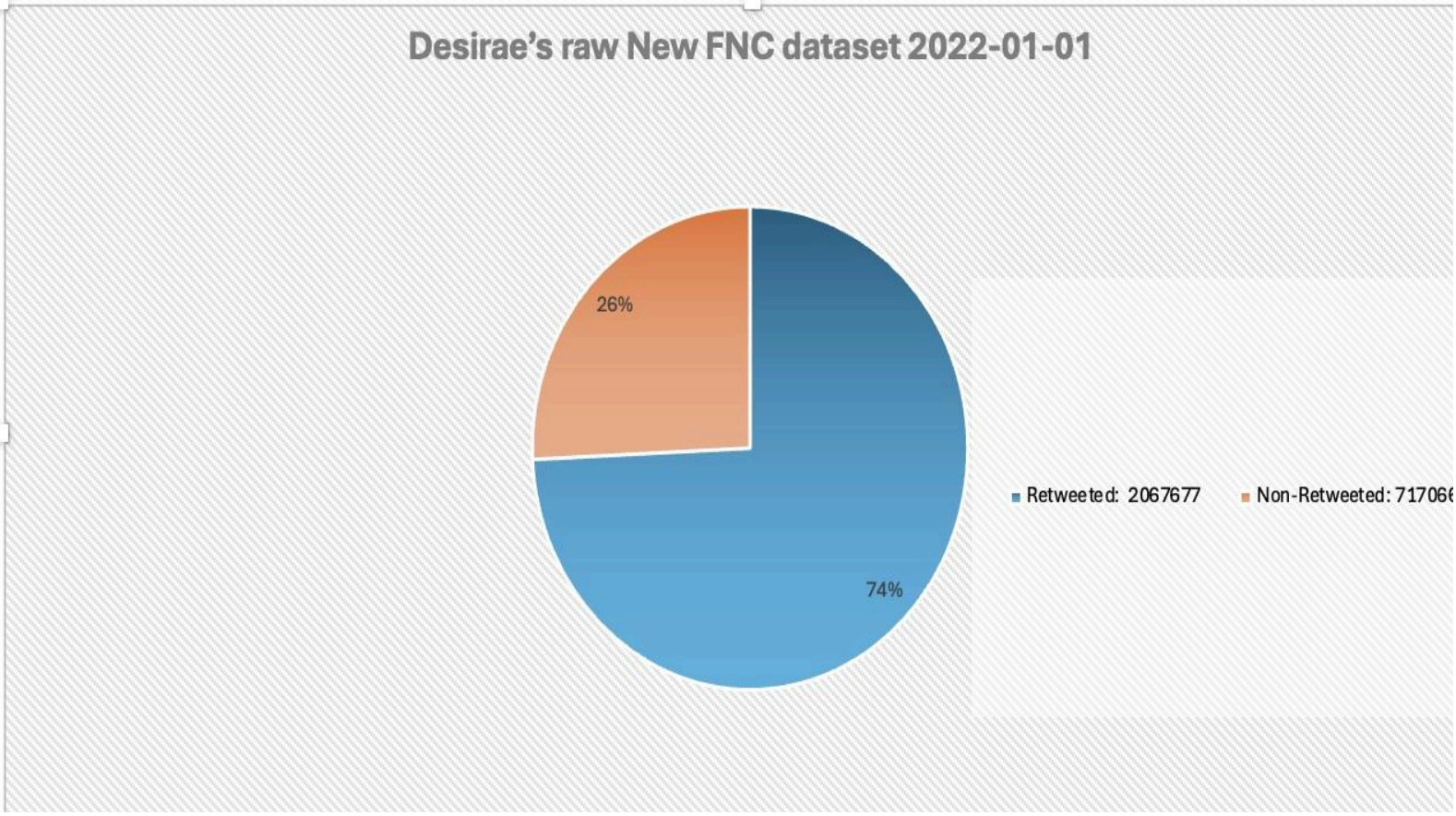

Figure 2: Desirae's New FNC Dataset Tweet Distribution

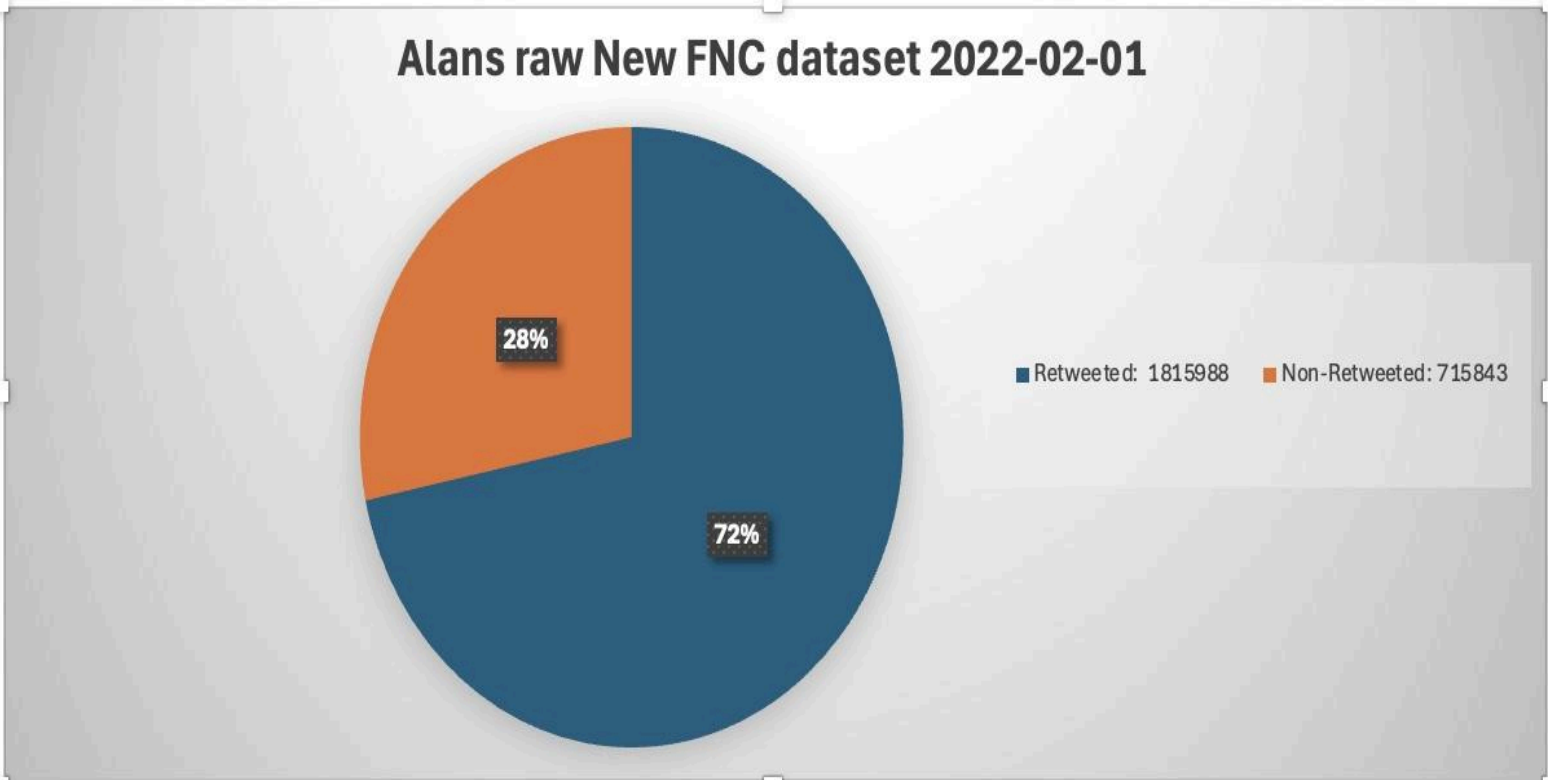

Figure 3: Alan's New FNC Dataset Tweet Distribution

These two graphs illustrate the distribution of retweeted and non-retweeted tweets in our raw New FNC datasets. Our analysis revealed that more than 70% of the data/tweets in both datasets consists of retweets. The high percentage of retweets highlights the repetitiveness of the New FNC datasets and the necessity of filtering to focus solely on original tweets.

We used the snorkel framework, which employs weak supervision, to label the FNC training datasets. We then used the resulting labeled data to train new expert models with the purpose of improving the accuracy rate of the pre-trained models by incorporating new recent insights of misinformation.

JSON Datasets

We previously mentioned the importance of the FNC Dataset's in our project, specifically the two New FNC Dataset's (January 2022 and February 2022). These two datasets were in JSON Format and were very noisy. Each raw New FNC Dataset had over hundreds of thousands of entries.


```
{
   "favorited":false,
   "in_reply_to_screen_name":null,
   "geo":null,
   "created_at":"Mon Jan 31 23:59:54 +0000 2022",
   "retweeted":false,
   "place":null,
   "retweet_count":0,
   "source":"<a href=\"http:\/\/twitter.com\/download\/android\" rel=\"nofollow\">Twitter for Android<\/a>",
   "is_quote_status":true,
   "user":{
      "lang":null,
      "id_str":"1057355098756661248"
   },
   "coordinates":null,
   "in_reply_to_status_id_str":null,
   "quoted_status":{
      "favorite_count":243,
      "id_str":"1488287928035983360",
      "retweet_count":35
   },
   "full_text":"A MAIOR MENTIRA DO MUNDO!!! Tem uns papudinho na minha rua que n\u00e3o pegam nem gripe!!! https:\/\/t.co\/n0XOjUo14W",
   "lang":"pt",
   "favorite_count":1,
   "in_reply_to_user_id_str":null,
   "id_str":"1488301044396343296",
   "id":1488301044396343296,
   "timestamp_ms":1643691594
}
```


Figure 3: New FNC Dataset in JSON Format

The image above shows one tweet entry, demonstrating the many fields in one entry. This complexity x the amount of entries, made filtering and cleaning the data challenging and required us to write Python scripts, which we will include later in our report.

A vital component of our project was the manual labeling of 500 twitter entries from both datasets, for a total of 1000 manual labeled entries. These 1000 manual labeled entries were labeled to establish the baseline of what constitutes as real and fake information within both the datasets. Using two separate datasets from different time periods was useful because

misinformation changes with time and we were able to take that into account.

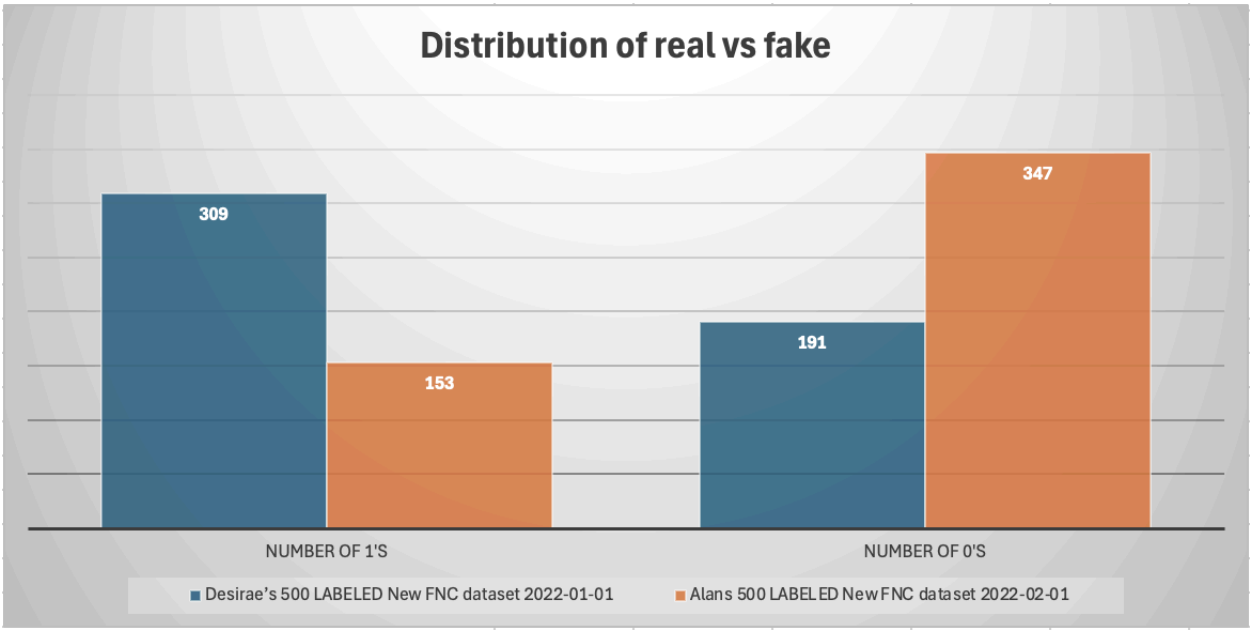

Figure 4: Real vs Fake in New FNC Dataset's

This bar chart compares the counts of real versus fake Twitter entries within each of our 500 manually labeled entries. The number of '1's represents the count of real entries, while '0's denote fake entries. This manual labeling, combined with weak supervision, provided a solid foundation for training and validating our models.

Method

The methodology describes the technique we used to identify and prevent the spread of COVID-19 misinformation. This includes various datasets, innovative machine learning models, and data processing.

Provided Dataset

The TA's gathered extensive datasets for training and evaluation from Twitter during the early months of the COVID-19 pandemic. This was compiled into a large JSON dataset which included thousands of Tweets. The Twitter data contained a variety of information and metadata which include user details, timestamps, and engagement metrics such as retweets and favorites. This data set was used because it reflects a time period where there would be the most relevant discussion about COVID-19 and the lockdowns. Therefore, the data set is essential for training a machine learning model to detect misinformation in the early COVID-19 landscape.

Annotation and Labeling

The annotation and labeling part of the KO project is important for good and accurate data. We first manually labeled 500 Tweets as real or fake news. This acts as a benchmark for future reference. In order to get the 500 Tweets, we had to filter out some Tweets. Non-english Tweets and retweets could not be used because it would be difficult to identify fake news in other languages since they have different grammar structures and nuances, and retweets would become repetitive and unhelpful for our goal. Below is a Python script ran to collect the 500 Tweets:

```python
import json

original_tweets = r'C:\Users\alanz\OneDrive - Georgia Institute of Technology\Documents\Georgia Tech\CS 4365\tweets-2022-02-01.json'
english_tweets = r'C:\Users\alanz\OneDrive - Georgia Institute of Technology\Documents\Georgia Tech\CS 4365\english-tweets.json'

english_tweet_list = []
count = 0

try:
    # Read the existing JSON file line by line
    with open(original_tweets, 'r', encoding='utf-8') as input_file:
        for line in input_file:
            if count == 500:
                break
            try:
                tweet = json.loads(line)
                if tweet.get("lang") == "en":
                    english_tweet_list.append(tweet)
                    count += 1
            except json.JSONDecodeError as e:
                print(f"Error decoding JSON: {e}")

    # Write the filtered tweets to a new JSON file with UTF-8 encoding
    with open(english_tweets, 'w', encoding='utf-8') as output_file:
        json.dump(english_tweet_list, output_file, indent=4, ensure_ascii=False)

    print("English tweets have been written to the new JSON file successfully.")
except PermissionError:
    print(f"Permission denied. Cannot read the file: {original_tweets}. Please check the file permissions.")
except Exception as e:
    print(f"An error occurred: {e}")
```

Figure 5: Python Script to Filter Language and Retweets

After creating a new JSON file with the 500 english only, non-retweeted Tweets, we converted this JSON file into a CSV file with the function provided by the Google Collab Jupyter Notebook:

Please upload the subsampled 500 tweets in the New FNC dataset to the "KO" directory in your Google Drive. The output from the weak supervision pipeline will also be there.

```
[ ] unlabeled_data_path = "/content/drive/My Drive/KO/english-tweets.json" # your subsampled JSON file
    labeled_data_path = "/content/drive/My Drive/KO/snorkel-date.csv" # the expected name of your output file
```

```
[ ] import pandas as pd
    unlabeled_data = pd.read_json(unlabeled_data_path, lines=False)
    unlabeled_data = unlabeled_data.loc[:, ["full_text"]]
    unlabeled_data = unlabeled_data.rename(columns={"full_text": "text"})
```

```
[ ] from snorkel.labeling import PandasLFApplier

    # Define the set of labeling functions (LFs)
    # change this to match the respective experts that you are using
    lfs = [lf_expert_albert_1_2020, lf_expert_bert_2_2020]

    # Apply the LFs to the unlabeled training data
    applier = PandasLFApplier(lfs)
    label_matrix = applier.apply(unlabeled_data)
```

Figure 6: Jupyter Notebook to Convert JSON to CSV

After creating the CSV file, we realized that some of the cells contained illegal characters for UTF, which prevented us from continuing to the next stage. These illegal characters were

likely emojis, so we had to run yet another Python script to clean the CSV, so that we could move on to the Snorkel section.

```
import pandas as pd

def filter_non_utf8(input_file, output_file):
    with open(input_file, 'r', encoding='latin1') as infile, open(output_file, 'w', encoding='utf-8') as outfile:
        for line in infile:
            utf8_line = line.encode('utf-8', errors='ignore').decode('utf-8')
            outfile.write(utf8_line)

input_file_path = 'C:/Users/alanz/OneDrive - Georgia Institute of Technology/Documents/Georgia Tech/CS 4365/2021-4.csv'
output_file_path = 'C:/Users/alanz/OneDrive - Georgia Institute of Technology/Documents/Georgia Tech/CS 4365/2021-4-clean.csv'

filter_non_utf8(input_file_path, output_file_path)

df_cleaned = pd.read_csv(output_file_path)
```

Figure 7: Python Script to Clean CSV

Finally, after this process, we obtained a valid CSV to begin labeling all 500 Tweets.

Weak Supervision and Snorkel

The weak supervision part of the project is the innovation for why the new machine learning model works. Snorkel allows the project to leverage imperfect heuristics, called labeling functions to produce labels for all the data. This prevents the need for manual annotation of large amounts of data. The following code snippets are the two labeling functions.

```
[ ]  @labeling_function()
     def lf_expert_albert_1_2020(x):
         """Follow this format to add other expert models"""
         model_config = copy.deepcopy(albert_config["model_param"])
         num_labels = albert_config["data_param"]["num_classes"]
         model_config["num_labels"] = num_labels
         model_config = AlbertConfig(**model_config)
         model = Albert(model_config).to(device)
         checkpoint = torch.load(albert_1_2020_checkpoint, map_location=device)
         model.load_state_dict(checkpoint['model_state_dict'])
         tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
         input = convert_to_features([x.text], [0], maxlen=tokenizer.model_max_length, tokenizer=tokenizer)
         all_input_ids, all_attention_mask, all_token_type_ids, all_input_len, all_labels = input[0]
         output = model(all_input_ids.to(device).reshape(1, -1), token_type_ids=all_token_type_ids.to(device).reshape(1, -1), attention_mask=al
         preds = torch.argmax(output.cpu(), dim=1)
         return int(preds)
```

Figure 8: lf_expert_albert_1_2020 function

```
[ ] @labeling_function()
    def lf_expert_bert_2_2020(x):
        """Follow this format to add other expert models"""
        model_config = copy.deepcopy(bert_config["model_param"])
        num_labels = bert_config["data_param"]["num_classes"]
        model_config["num_labels"] = num_labels
        model_config = BertConfig(**model_config)
        model = Bert(model_config).to(device)
        checkpoint = torch.load(bert_2_2020_checkpoint, map_location=device)
        model.load_state_dict(checkpoint['model_state_dict'])
        tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
        input = convert_to_features([x.text], [0], maxlen=tokenizer.model_max_length, tokenizer=tokenizer)
        all_input_ids, all_attention_mask, all_token_type_ids, all_input_len, all_labels = input[0]
        output = model(all_input_ids.to(device).reshape(1, -1), token_type_ids=all_token_type_ids.to(device).reshape(1, -1), attention_mask=al
        preds = torch.argmax(output.cpu(), dim=1)
        return int(preds)
```

Figure 9: lf_expert_bert_2_2020 function

Model Training

The model training and refinement phase is where the expert model is trained for misinformation detection based on the labeled data from the previous section. The two models provided were the BERT and ALBERT which were chosen for their natural language processing ability. Here are the two functions.

```
[ ] """Adapted from https://github.com/huggingface/transformers/blob/v4.34.0/src/transformers/models/albert/modeling_albert.py#L1025"""
    import torch.nn as nn
    from transformers import AlbertModel, AlbertPreTrainedModel

    class Albert(AlbertPreTrainedModel):
        def __init__(self, config):
            super().__init__(config)
            self.config = config
            self.encoder = AlbertModel.from_pretrained("albert-base-v2", config=self.config)
            self.dropout = nn.Dropout(self.config.classifier_dropout_prob)
            self.classifier = nn.Linear(self.config.hidden_size, self.config.num_labels)

            # Initialize weights and apply final processing
            self.post_init()

        def forward(self, x, attention_mask=None, token_type_ids=None, position_ids=None, head_mask=None):
            outputs = self.encoder(x,
                                   attention_mask=attention_mask,
                                   token_type_ids=token_type_ids,
                                   position_ids=position_ids,
                                   head_mask=head_mask)

            pooled_output = outputs[1]
            pooled_output = self.dropout(pooled_output)
            logits = self.classifier(pooled_output)
            return logits
```

Figure 10: Albert Model

```
[ ]  """Adapted from https://github.com/huggingface/transformers/blob/v4.34.0/src/transformers/models/bert/modeling_bert.py#L1519"""
     import torch.nn as nn
     from transformers import BertModel, BertPreTrainedModel

     class Bert(BertPreTrainedModel):
         def __init__(self, config):
             super().__init__(config)
             self.config = config
             self.encoder = BertModel.from_pretrained("bert-base-uncased", config=self.config)
             classifier_dropout = (
                 self.config.classifier_dropout if self.config.classifier_dropout is not None else self.config.hidden_dropout_prob
             )
             self.dropout = nn.Dropout(classifier_dropout)
             self.classifier = nn.Linear(self.config.hidden_size, self.config.num_labels)

             # Initialize weights and apply final processing
             self.post_init()

         def forward(self, x, attention_mask=None, token_type_ids=None, position_ids=None, head_mask=None):
             outputs = self.encoder(x,
                                    attention_mask=attention_mask,
                                    token_type_ids=token_type_ids,
                                    position_ids=position_ids,
                                    head_mask=head_mask)

             pooled_output = outputs[1]
             pooled_output = self.dropout(pooled_output)
             logits = self.classifier(pooled_output)
             return logits
```

Figure 11: Bert Model

Of these two models, we had better luck using the Albert model as it ran much quicker and gave us better results. This was the model we used for all of our datasets. Once the models are refined and show good performance, they are then integrated back into the Snorkel pipeline. This ultimately provides accurate and reliable misinformation detection.

Evaluation

After the model is created, we must evaluate the effectiveness and accuracy of the misinformation detection on the models. The metrics we used to measure effectiveness were precision, which is the proportion of true positives of all positives, recall, which was the proportion of true positives of all actual positives, F1 score, the harmonic mean of precision and recall which gives a metric for both aspects, and accuracy which is the measure of all correct predictions of all predictions made. This can help us measure the quality of our model.

Schedule Breakdown

This section breaks down the tasks for each week and delegates the responsibilities for each task. Note that for the first week, we initially decided to do a student proposed project instead of the structured KO project, but then decided to switch to the structured KO project. Because of this change, we lost a week of progress, so we had to catch up for weeks 3 and 4. Ultimately, this did not affect our schedule by much.

Sprints	**Tasks**
Week 1-2	● Introduction to the project and proposal document (Alan & Desirae) ● Reconsider switching student proposed to structured project (Alan & Desirae) ● **Deliverable:** Join private repo
Week 3-4	● Introduction to the project and proposal document (Alan & Desirae) ● Annotate the new FNC dataset assigned to each individual (Alan) ● Setup environment and run demo tasks on scripts (Desirae) ● **Deliverable:** Data annotation
Week 5-6	● Train models on early/previous months and test on later/future months (Alan & Desirae) ● **Deliverable:** Test accuracies
Week 7-8	● Use Snorkel to assemble NELA and old FNC models (Alan) ● Train new models on earlier new FNC and test on later new FNC (Desirae) ● **Deliverable:** Experiment statistics
Week 9-10	● Use Snorkel to assemble early models (Alan) ● Test on later new FNC (Desirae) ● **Deliverable:** Submit deliverables, report, and recorded video (Alan & Desirae)

Results

After creating our model and running performance metrics for it, we were able to generate the following plot to measure the test accuracies for the NELA and Old FNC models using the ALBERT model.

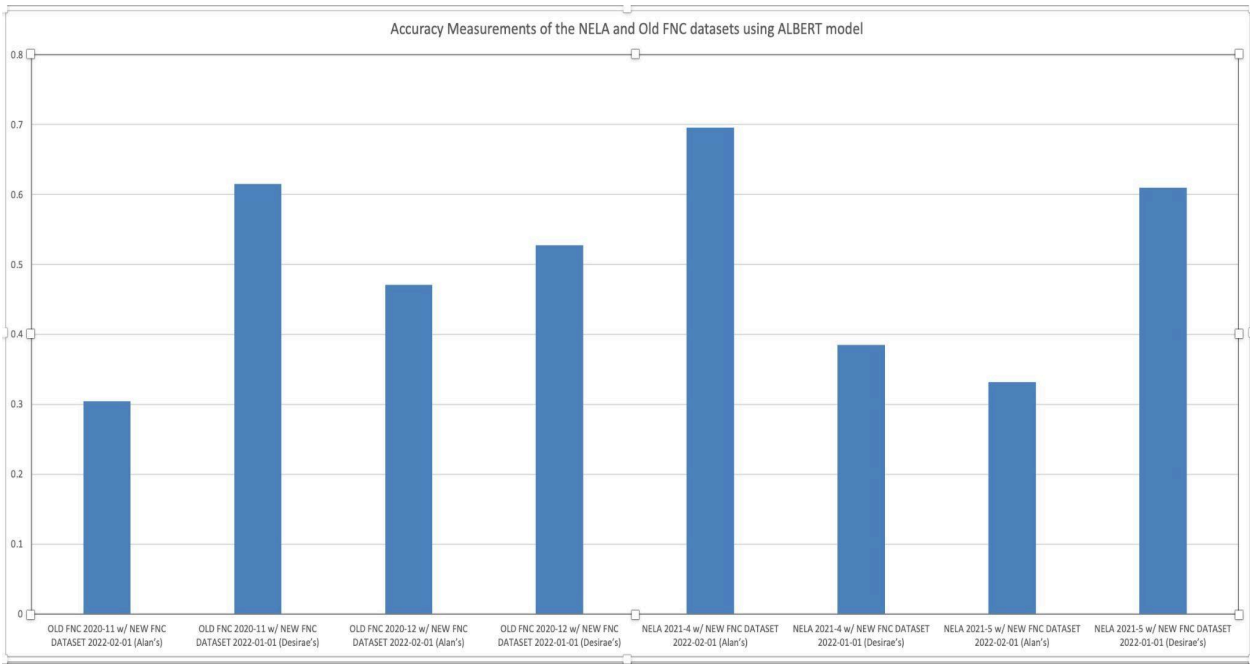

Figure 12: Accuracy Measurements

This bar chart displays the accuracy percentages from each of the four experiments, for a total of 8, conducted in Follow-up #2. These experiments evaluated the accuracy of the NELA and old FNC datasets using models trained on our labeled datasets. Additionally, there is a graph that compares the performance results of the models assembled using Snorkel for training and testing on new FNC models.

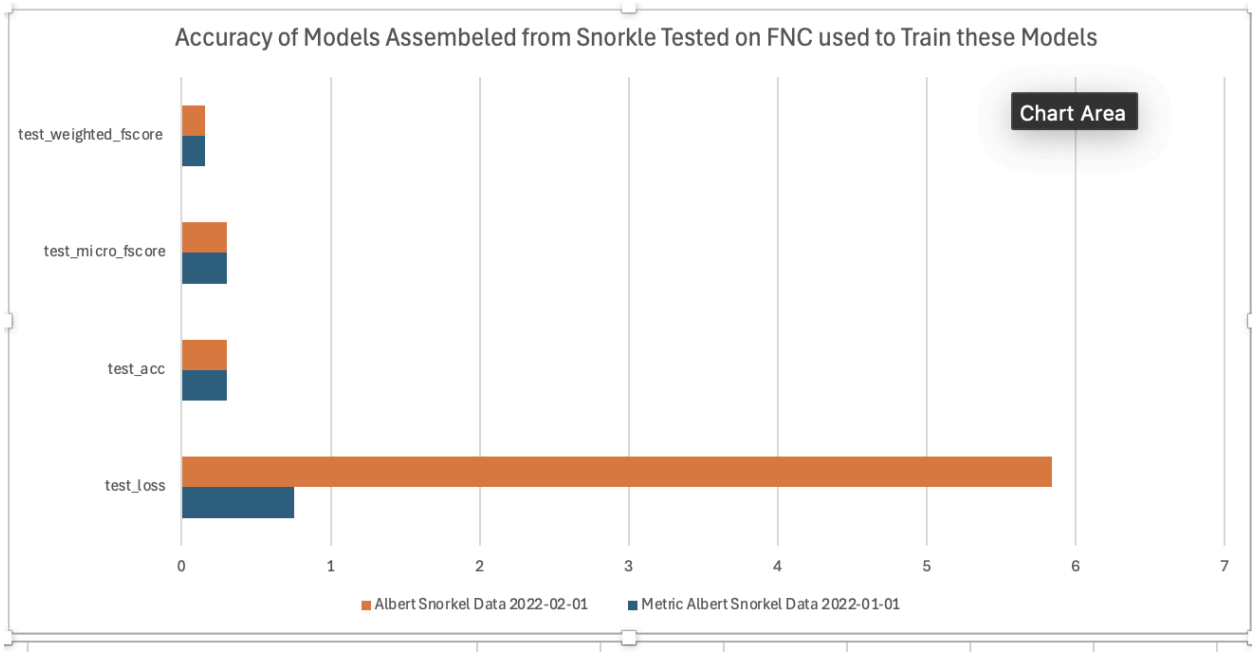

Figure 13: Accuracy of Models Assembled from Snorkel

In order for our model to be accurate, it must correctly label a datapoint as fake or real news accurately more than random chance. Unfortunately, our model did not fit that description, so some more training is required to create a more accurate model. Our current model kept return positives for every data set, so there is a possibility something went wrong with the training process which resulted in this error. The following confusion matrix displays this error.

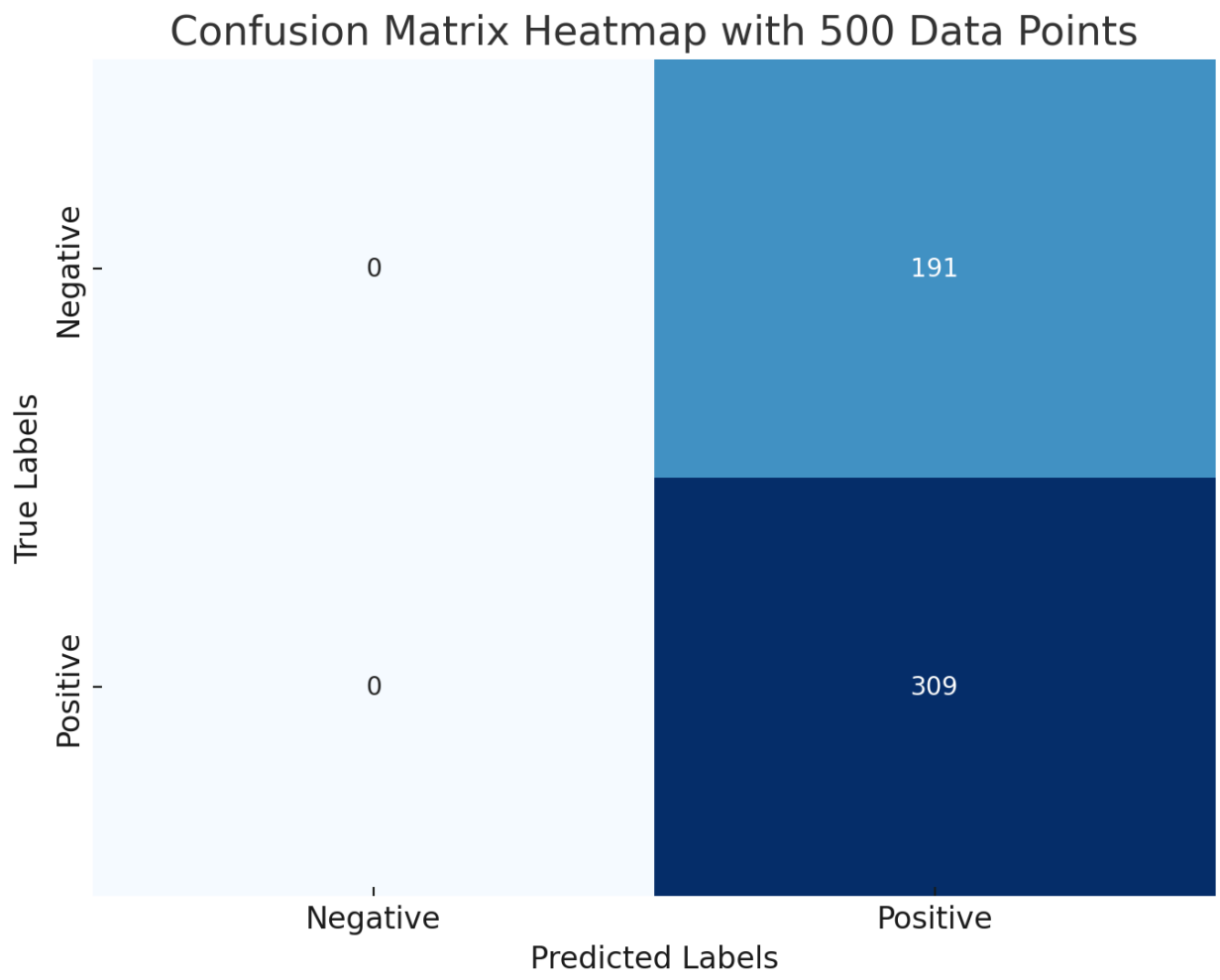

Figure 14: Confusion Matrix Heatmap

Future Research

There are several ways that the concept of a dynamic and adaptable machine learning model could be used for future technologies. This innovation has proven that it is possible for machine learning algorithms to keep up with the fast and ever evolving pace of new and current data. Whether its more robustness or overall refinement, this technology can always be improved. For the future, new types of data and data sources can be explored. For example, instead of focussing primarily on COVID-19 data or even health data, researchers could pivot to

other relevant news like natural disasters. Any data stream that acts in real time can use this methodology. Additionally, other machine learning techniques could be employed instead of just BERT and ALBERT. This may greatly change the test accuracies of the models. Finally, the models could be scaled into cloud implementations or other integrations where the technology can be found to be useful. Overall, this research is still very new and can be researched and developed into new applications, though it is important to consider the ethics of real time data streams, such as privacy and security.

Skills Learned

Before officially beginning the Knowledge Obsolescence project, our team created a list of skills we aimed to learn or improve throughout its completion. These skills included Data Annotation, Machine Learning, Programming, Model Evaluation, and Project Management.

Skills/Goals

Data Annotation:

- Experience with labeling datasets for machine learning models.
- Ability to identify and classify misinformation accurately.

Machine Learning:

- Understanding of weak supervision techniques.
- Knowledge of training and fine-tuning models.
- Familiarity with Snorkel for integrating multiple labels.

Programming:

- Proficiency in Python or other relevant programming languages.
- Experience with data manipulation and preprocessing libraries (e.g., Pandas, NumPy).

Model Evaluation:

- Skills in evaluating model performance using various metrics.
- Ability to compare different models' performance and make informed adjustments.

Project Management:

- Experience with managing sprints and tasks in a collaborative environment.
- Ability to document progress and report on project status.

Figure 15: Initial Skills Listed, Follow Up 1

As we progressed, we tracked our skill development, making sure to note improvements and reflect on how and why we advanced. This process helped us stay aware of our growth and maintain a record of what we are learning.

With this being said, Follow-Ups 2 and 3 were used to act as a timeline of our progress, which we will include as well. These follow ups provide a detailed summary of how our skills evolved over the course of the project at various stages.

Learning skills report:

A few skills strengthened and/or obtained in this phase of the project include:

Data Annotation: By the end of follow up 2 for the KO project, we can say with certainty we have fully learned this skill. In Follow up 1, we only made progress towards this skill, but we mastered it here as we completed another round of annotating an additional 500 data entries totaling up to 1000. We annotated the data based on what we believed to be true or false using 0's and 1's, with the purpose of using this annotated data to train our ML models.

Machine Learning: We are continuing to make strong progress in learning this skill. We say this because, we now have mastered the concept of weak supervision, as we provided/completed an additional 500 entries of labeled data to assist in training our models which is a large part of Machine Learning. We also trained our models on earlier months and tested on later months.

Programming: After completing follow up 2 for the KO project, we can also add that we have moved another step in mastering programming, specifically programming to achieve data manipulation. I say we have moved another step because there is no final learning point when it comes to coding and we are sure we will write more code in the future, whether in or outside this class, to assist us. Our Python script that we used successfully extracted English words and non-retweets into a new JSON file, assisting us in our labeling process. Model Evaluation: We have made slight progress with this skill. We say this because in this phase of the project, we trained our models and tested for accuracy. Now we can use this to begin assembling our NELA and old FNC models.

Project Management: Our group is one step closer to mastering project management. Right now, we say we have learned a great bit on the topic and have been successful so far. We say this because we successfully conducted 8 experiments for our KO project, completed our own individual assignments, as well as completed our follow-up before the due date. We also made sure that all our deliverables for this phase were up to par and in great standard. We still have much to learn as the next phases will require more work while having the same time frame to complete.

Figure 16: Skills post Follow-Up 2

Learning skills report:

A few skills strengthened and/or obtained in this phase of the project include:

Data Annotation: Our team decided that in follow up 2 we have mastered this skill but would like to extend on this for follow up 3. We participated in Data annotation in this phase of the project as well, however instead of us manually inputting our annotations, we used Snorkel. Therefore, we have fully learned this skill in both manually annotating as well as using technologies such as Snorkel to do this for us.

Programming: After completing phase 3 for the KO project, our group can say that we have gained a large amount of experience in using Python to write code to not only manipulate data, but to get information from data as well. In follow up 3, to create a few of our graphs we needed to get information from our datasets that would have taken too long to search for manually. So, we wrote python scripts to find information such as the number of 1s and 0s in our labeled datasets and the number of retweets and non-retweets in our datasets. With this being said we are one step closer to mastering this skill.

Model Evaluation: We have made good progress in mastering this skill. We say this because we have successfully assembled our NELA and old FNC models and tested the accuracy of our machine learning models using annotated datasets. From our trainings and testing's, we were able to create graphs of our findings.

Project Management: Our group is very close to mastering project management and proof of this is that we have learned from our initial hiccup and have been able to successfully complete each assignment and follow up to the best of our abilities. For follow up 3, our group produced 4 graphs of data we deemed important to know from our KO project. We put a lot of time and effort into our graphs and are happy with how they came out. We also were able to successfully complete our own individual assignments before the due date. We will not say we have completely mastered this skill until we have successfully submitted all our final deliverables including the presentation, video, and project files.

Figure 17: Skills post Follow-Up 3

References

Abhijit Suprem and Calton Pu “The New-Normal of Fake News: Enhanced and Sustainable Multi-Expert Detection through Timely Adaptation to Counter Knowledge Obsolescence” https://asuprem.com/static/content/koinitial.pdf

Berrondo-Otermin, Maialen, and Antonio Sarasa-Cabezuelo. 2023. "Application of Artificial Intelligence Techniques to Detect Fake News: A Review" Electronics 12, no. 24: 5041. https://doi.org/10.3390/electronics12245041

Zhang, Litian, Xiaoming Zhang, Ziyi Zhou, Feiran Huang, and Chaozhuo Li. 2024. “Reinforced Adaptive Knowledge Learning for Multimodal Fake News Detection". Proceedings of the AAAI Conference on Artificial Intelligence 38 (15):16777-85. https://doi.org/10.1609/aaai.v38i15.29618.