

B.Comp. Dissertation

Online platform for temporal claim and evidence labelling

By Ye Jiadong

Department of Computer Science

School of Computing

National University of Singapore

2022/2023

Project Number: H037310

Advisors: Prof. Lee Mong Li Janice (SOC), Prof. Wynne Hsu (SOC)

Co. Advisor: Prof. Teo Chung Piaw (BIZ)

Deliverables:

Report: 1 Volume

Abstract

Fake news has become increasingly prevalent over the past decade with the proliferation of social media networks. One of the most pressing challenges in this area is to develop automated fact verification systems that are highly accurate in identifying the factual truths of statements. An important part of such a system would involve a machine learning classifier model that compares a claim statement with retrieved evidence sources and makes a final decision on whether the evidence sources “support”, “refute” or do not possess sufficient information to verify a claim. Challenges include being able to recognise both the general and temporal elements of claims, and gathering sufficient training data to ensure a high classification accuracy. It is often also difficult to evaluate the performance of such classifier models, due to the lack of annotated fact-evidence datasets with high quality annotation. In our work, we have designed an annotation system that makes use of the newly released GPT transformer models to support both annotators as well as investigators in ensuring a relatively high level of annotation quality. We have also proposed a pipeline that allows these annotations to be fed back to the GPT system to train a fact-verification classifier “on the fly” as annotation is being carried out. Our machine learning models are evaluated based on accuracy metrics on a separate unseen test dataset. The ability of our annotation system to improve annotation quality will be gauged based on an comparative user test carried out by an externally engaged annotator.

Subject Descriptors:

D.2. Software Engineering

I.2. Artificial Intelligence

Keywords:

Fact verification, annotation platforms, transformer architectures, GPT

Major Implementation Software and Hardware:

Vue JS, Django, MySQL, HuggingFace, Paperspace Gradient, GPT

Acknowledgement

Despite the challenges of managing a dual business and computer science thesis, I want to thank my respective supervisors from computing Prof. Janice Lee and Prof. Wynne Hsu for their understanding and guidance through our initial discussions, as well as Prof. Teo Chung Piaw from Business school for his inputs on the predictive analytics elements of the thesis. I would also like to thank Anab, who has been highly instrumental in providing me with initial sample data, and also giving me invaluable inputs on the annotation process. Lastly, I also want to thank my evaluator from computer science, who has given me mid-term inputs that have prompted me to think about my research methodology, and deliver a convincing result. This is the final project that I will complete as an undergraduate student, and despite it being the toughest, it will be one that I will remember for the next phase of my own journey.

Table of Contents

1. Introduction and Background	6
1.1 Problem (Fake News)	6
1.2 Fact Verification	6
1.3 Automated Fact Checking	7
1.4 Temporal vs Non-Temporal Claims	7
2. Existing fact verification datasets	7
2.1 Existing Annotation Review Mechanisms	9
3. Problem Statement	10
4. Objective	11
5. Research Methodology	11
6. Literature Review	12
6.1 Non-Transformer Architectures	12
6.2 Transformer Architectures	13
6.3 Transformers for Text Classification	14
7. Model Implementation and Results	15
7.1 Dataset	15
7.2 Preprocessing	16
7.3 Transformers	17
7.4 Additional Feature Extraction	20
7.5 Classifier Model	23
7.5.1 Decision Tree, Random Forest, Gradient Boosting	23
7.5.2 Support Vector Machine	26
7.5.3 Neural Network	27
7.6 Generative Pre-Formed Transformers (GPT)	29
7.6.1 Prompt-Based Tuning	29
7.6.2 Non Prompt-Based tuning	32
7.7 Final Model Evaluation and Selection	32

8. Web-based annotation platform	34
8.1 Design Decisions	35
8.1.1 Overall Pipeline	35
8.1.2 Database	36
8.1.3 Backend Infrastructure	38
8.1.4 Frontend Infrastructure	38
8.2 Overall Architecture	38
8.3 Tech Stack	39
8.4 Evaluation	39
9. Conclusions	40
9.1 Summary	40
9.2 Limitations	40
9.3 Recommendations for Future Work	41
References	42
Appendix A – Final Prompt used for GPT 3.5	45
Appendix B – Annotation Platform	47
Administrator Dashboard	47
Annotator	48
Stepwise Flow (With UI)	49
GPT’s role in highlighting temporal and relevant elements	50
Manual Review	51

1. Introduction and Background

1.1 Problem (Fake News)

Fake news is defined as news articles that are intentionally misleading and verifiably false, often spread for nefarious purposes. Fake news has become increasingly prevalent in the digital age, especially when people around the world increasingly turn to social media for news. In the United States, close to 50% of respondents surveyed by the Pew Research foundation rely on social media platforms such as Facebook, Youtube, and Twitter for news (Atske, 2021). Many of these platforms also lack safeguards against the limitless dissemination of unverified information, which can spread across inter-connected social networks at an exponential pace. In fact, a single piece of fake news can reach a global audience within mere minutes (Lee & Raviprakash, 2021). Coupled with the information overload, it is challenging for people to discern fake news. A survey carried out in Singapore with 750 respondents showed that over 90% of respondents mistakenly identified fake news headlines as being real (Ipsos, 2018). The consequences of fake news are often also severe, and wide-ranging. Socially, it can intensify social conflict (Centre for Information Technology & Society). Businesses should also be worried, as fake news has the potential to negatively impact share prices and shareholder sentiments (Atkinson, 2019).

1.2 Fact Verification

Due to the destructive and far-ranging consequences of fake news, it is important to combat fake news. It is difficult, and often impossible to prevent the generation of fake news. Instead, we focus on fact verification, which is an integral part of the process to prevent the spread of fake news. Manual fact verification involves carefully scrutinising every detail of news articles, which is what journalists in major reputable news agencies do. On social media platforms, due to the volume of information being shared, manual fact checking relies on user reports, followed by a laborious process of moderation staffed by “trust and safety” or “content moderation” teams. However, manual verification is often too slow, and cannot keep up with the sheer volume of content that is shared (Zeng et al., 2021). Instead, we focus on automated fact checking, which not only reduces human manpower required, but also the time taken to sieve through each informational source before it is disseminated to the population.

1.3 Automated Fact Checking

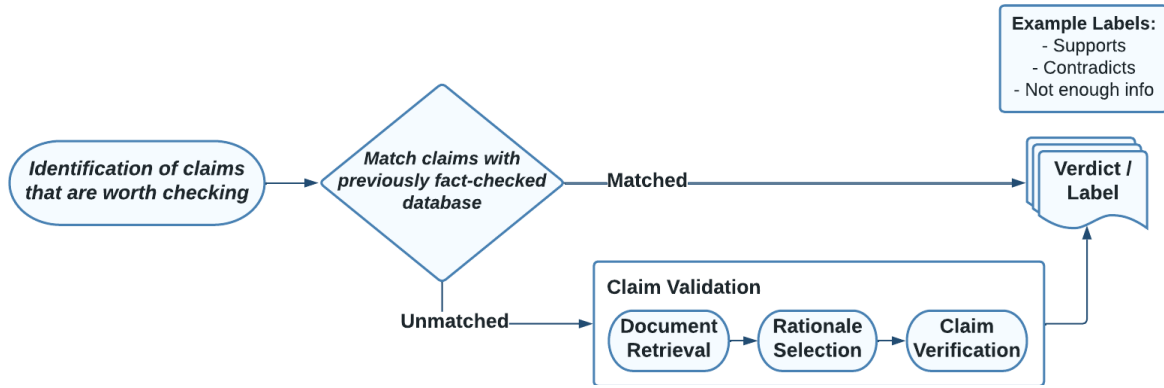


Figure 1 - Automated Fact Checking Process (Zeng et al., 2021)

The automated fact checking process begins with the identification of claims that are worth checking. Some claims might not be factual in nature (i.e. I want to eat ice-cream) and contain only opinions. These do not need to be fact-checked. Then, identified claims are matched with previously fact-checked databases to see if a verdict can be generated immediately. If the claim has not been “seen” before, then it proceeds to the claim validation phase, which involves the use of machine learning models to verify claims against retrieved documents from publicly available sources (i.e. Wikipedia, news agencies). The machine learning model gives a final verdict in the form of predefined labels such as “SUPPORTS” or “REFUTES”, together with identified parts of the retrieved documents that serve as the rationale for the claim’s final verdict. In order to train and assess the performance of machine learning models to verify claims and predict labels, large scale annotated datasets are required.

1.4 Temporal vs Non-Temporal Claims

Claims can be differentiated into temporal and non-temporal forms. Temporal claims contain time information. For example, “Boris Johnson resigned as the UK prime minister in 2022” is a temporal claim as it contains the time-related “in 2022” phrase. Automated fact checking now involves an additional level of complexity for temporal claims, as the temporal elements of a claim have to be verified together with the facts of the claim. In order to construct machine learning models to automatically predict fact labels, there needs to be a sufficiently large annotated dataset that these models can be trained, and evaluated on. In the next section, we review some of the existing popular datasets used.

2. Existing fact verification datasets

There are several existing datasets for non-temporal fact extraction and verification. These datasets are manually annotated, and the majority contain only labelled claims and evidence,

together with their final labels. All of these datasets do not distinguish temporal from general elements of both claims and evidence, and include only a final truth label. In addition, some datasets such as the Liar Dataset constructed in 2017 adopted a feature driven approach, by including additional features of the claim and evidence that could influence the final label.

Column(s)	Description
id	The json ID of the statement
label	Truth value of the statement; 6 categories from 'true' to 'pants on fire'
statement	Title of the PolitiFact article, often but not always the actual statement
subject	The subject(s) of the statement
speaker	The source of the statement
speaker_job speaker_us_state speaker_affiliation	The speaker's job title, US state where they're based, and party affiliation, where available
speaker_bt ... speaker_pof (5 features)	Total count of truth values for the speaker (truth credit history), excluding 'true' count and including the current statement
context	The context (venue / location of the speech or statement)

Figure 2 - Liar Dataset Final Output (Ouyang, 2020)

However, this approach poses a strain on the annotation process. Without a proper annotation review process, the annotator will have to sieve out all of these key features from different sources, increasing the risk of annotation error. It is also unclear which of these features really influence the final label, leading to decreased annotation efficiency when annotated features do not directly influence the final label. Including features which might not have a significant contributory impact on the final label can potentially also detriment the quality of training done using the dataset.

We then explore another huge dataset - the FEVER, which captures close to 185000 claims from Wikipedia content. In order to construct the dataset, a similar annotation tool shown below was utilised.

Claim Labelling Task (WF2)

Claim: Barbara Bush was a spouse of a United States president during his term.

Buttons: Submit, Submit and flag, Skip (opens menu), Home, Guidelines

Wikipedia article for Barbara Bush

Barbara Bush (née Pierce; born June 8, 1925) is the wife of [George H. W. Bush](#) , the 41st President of the United States , and served as First Lady of the United States from 1989 to 1993. ✓ Supports ✗ Refutes Cancel

She is the mother of [George W. Bush](#) , the 43rd President, and [Jeb Bush](#) , the 43rd Governor of Florida . Expand

She served as the [Second Lady of the United States](#) from 1981 to 1989. Expand

Barbara Pierce was born in Flushing, [New York](#) . Expand

She attended Milton Public School from 1931 to 1937, and Rye Country Day School from 1937-1940. Expand

Instructions:

Add a custom page from Wikipedia if essential information is missing from the dictionary. E.g. the claim mentions an entity that does not appear in the Wikipedia page for Barbara Bush. Add Custom Page

If you need to combine multiple sentences from the original page (Barbara Bush), this will add it to the dictionary so that it can form part of the supporting evidence. Add Main Wikipedia Page (Barbara Bush)

Quick Links

- [First Lady of the United States](#)
- [George H. W. Bush](#)
- [George W. Bush](#)
- [List of Presidents of the United States](#)
- [First Lady of the United States](#)

☐ First Lady of the United States (FLOTUS) is the informal but accepted title held by the wife of the President of the United States, concurrent with the president's term of office.

Figure 3 - FEVER Labelling Task (Thorne et al., 2018)

However, as FEVER was not designed specifically considering temporal claims, temporal labels were not specifically assigned. An overall label of “Supports”, or “Refutes” was given by annotators based on the consideration of the entire claim and evidence sentences. Hence, it was not possible to tell if the claim was refuted because of a mismatch of time information present in the claim, or due to non-factual elements of the claim. Additionally, the annotation platform was difficult to use. Figure 3 shows the cluttered layout with multiple panels on both the left and right sides of the screen. Annotators were looking at multiple pieces of evidence on the same screen, in order to come to a conclusion on the claim, further increasing the risk of annotation errors.

2.1 Existing Annotation Review Mechanisms

Besides the use of decluttered and easy-to-use user interfaces for the annotation process, as well as ensuring clear annotation guidelines (Vidgen & Derczynski, 2020), annotation review mechanisms are necessary to ensure a high standard of annotation quality. For annotating significantly large datasets, it is impossible for the investigator or researcher to sieve through all annotated entries. Current review mechanisms include looking at inter-rater agreement, or simply taking the majority label from all annotators who annotate the same entry. Inter-rater agreement is calculated using statistical measures such as the Fleiss kappa, which has to be computed for the entire dataset. This means that even if the inter-rater agreement indicates low agreement, it is difficult to sieve out specific entries that should be discarded. Instead, investigators often discard the entire annotated dataset. On the other hand, majority measures

are unreliable in borderline situations. For example, if 30 annotators disagree, while 31 annotators agree with one another, it is unclear if that one additional annotator’s input can and should decide the majority. Lastly, investigators often randomly sample a subset of the annotation entries (i.e. 10%), and mark the entries manually to get an idea of the annotation quality. Again, since investigators often have incomplete information about the population of their annotators, the selected samples might not be representative of all annotations done.

3. Problem Statement

With the existing review mechanisms and datasets available discussed, we find that there are no existing annotation systems that are able to generate temporal fact-verification datasets with a structured annotation review process that can ensure annotation quality is not compromised. As a result, annotation errors are often commonplace in these datasets, and further degrade classifier performance. A MIT study has found that major machine learning datasets are prone to significant annotation errors, on an average of 3.4% across all datasets (Conner-Simons, 2021). This can pose a challenge to downstream models that depend and train on the annotated data. Research done that looked at annotation errors on the accuracy of convolutional neural networks (CNN) concluded that an additional 10% of noise led to a 4% reduction in the CNN’s accuracy (Flatow & Penner, 2017).

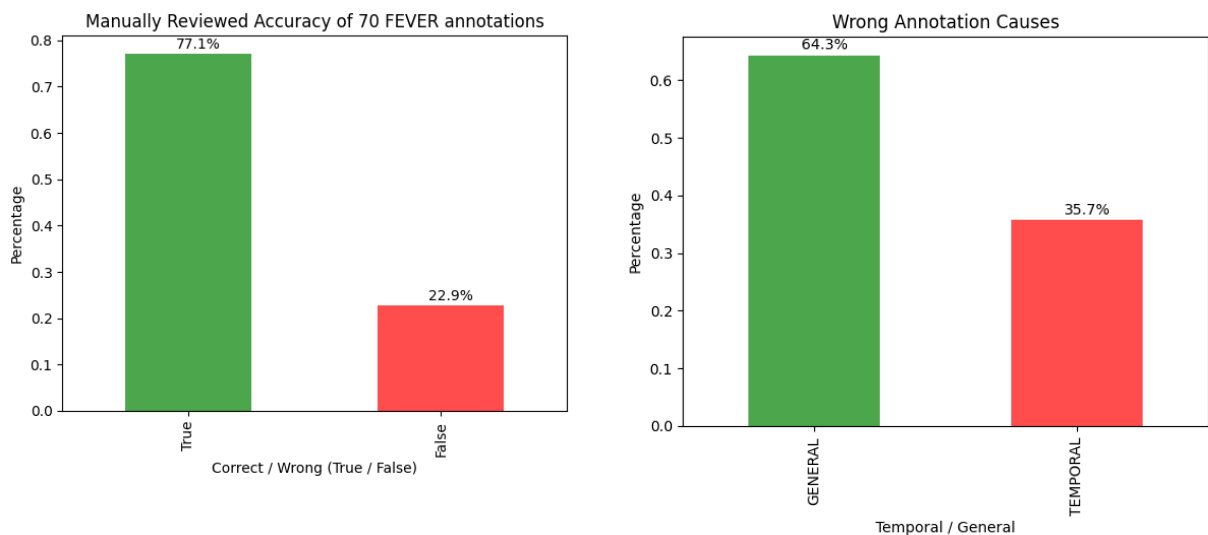


Figure 4 - Annotation errors in the FEVER dataset

We manually reviewed 70 previously annotated entries of the fact-verification FEVER dataset, which were all given the “SUPPORTS” label. We found that the annotation accuracy was only 77.1%. 16 out of the 70 annotations done were incorrect, and should have been

“NOT ENOUGH INFO”. We also attributed reasons for the incorrect annotation and found that in 64% of wrong labelling, it was due to missing general information within the evidence statement. In 35% of wrong labelling, it was due to missing temporal information. Clearly, both temporal and the general parts of a claim are important in determining truth. We give an example of an incorrect annotation done below:

Claim: George H. W. Bush was a Republican politician born June 12, 1924.

Evidence: A member of the Republican Party, he was previously a congressman, ambassador, and Director of Central Intelligence.

Annotated Label: “SUPPORTS”

The annotator has failed to verify the temporal elements of the claim, and the evidence cannot be seen as supporting the claim if it fails to address a key tenet of the claim, which is that the individual was “born June 12, 1924.”.

4. Objective

Our project aims to design and develop an web-based platform for temporal claim and evidence labelling, while deploying a predictive model to assist annotators and serve as automated annotation review. This seeks to improve annotation quality, and improve annotation efficiency through reduced manual reviews. Ultimately, good quality annotations done on the platform can be used in the construction of a robust large-scale temporal dataset containing annotated claims and evidence, that serves to further improve and evaluate fact-verification classifier models.

5. Research Methodology

For our web-based platform, we took inspiration from the PathoJen project team (Hahn et al., 2012), who designed an active learning system to label medical corpuses. PathoJen’s key objective was to reduce annotation costs without compromising on annotation quality by introducing a “sampling bias”. A predictive model was used during annotation to assign only entries that are comparatively more difficult to annotate to the human annotators, while predicting the easier annotations itself. Over time, it was also able to learn and improve from its previous mistakes with the human annotations. We decided to adapt this architecture, but give the model a different role. Our annotators have to annotate how a given evidence sentence would verify a specific claim. In addition to the annotator’s annotation, our predictive model makes its own prediction on the truth value, and its output is compared to determine if manual reviews are required. In order to evaluate our system, we focused on two

fronts. The predictive model has to be at least within 10 percentage points of an annotator’s accuracy on an unseen test set. We know from our manual checks on the FEVER dataset that annotation accuracy is around ~70%, so anywhere around ~60% would be reasonable. In order to determine if the annotation system benefits annotators, a user test of the annotation system will be carried out to determine if annotation entries done through our platform were of a higher accuracy.

6. Literature Review

We first explored existing natural language architecture that would be suitable for a fact-checking classifier.

6.1 Non-Transformer Architectures

Model	How does it work?	Pros	Cons
Bag of Words (BOW)	Counts how many times different words appear in the text corpus and generates a vector of different word frequencies	<ul style="list-style-type: none"> - Extremely simplistic and easy to generate - Efficient even for large datasets 	<ul style="list-style-type: none"> - Does not factor in semantics (meaning) - Most frequent word does not mean most relevant
TF-IDF (Term frequency-inverse document frequency)	Similar to the BOW model, TF-IDF attempts to look at the significance of each word in a text document. However, it offsets this by how often the word appears across the entire text corpus.	<ul style="list-style-type: none"> - Addresses the BOW’s downside of not being able to deal with common stopwords (i.e. “a”, “the”) - Considers the relative importance of words 	<ul style="list-style-type: none"> - Context agnostic (as with BOW model) - Different document lengths can give varying results, leading to incomparable results

Recurrent Neural Networks (LSTMs, GRUs, Bidirectional)	Uses sequential neural networks to understand word context and relevance in a sentence	<ul style="list-style-type: none"> - Able to capture contextual features - Sequential processing allows it to capture how the order of words influence the meaning of a sentence - Possess memory, and able to adjust current decision based on previous information 	<ul style="list-style-type: none"> - Computationally very intensive - Are not able to focus on specific elements of input text that might be more important due to sequential processing
---	--	---	--

Table 1: Comparison of the different NLP models (Chiusano, 2022)

6.2 Transformer Architectures

Transformers were first popularized in 2017 by Google researchers. They present a simpler “network architecture” that focuses on “self-attention” mechanisms (Ashish Vaswani et al., 2017), yet provides performance and accuracy gains over conventional and traditional neural networks.

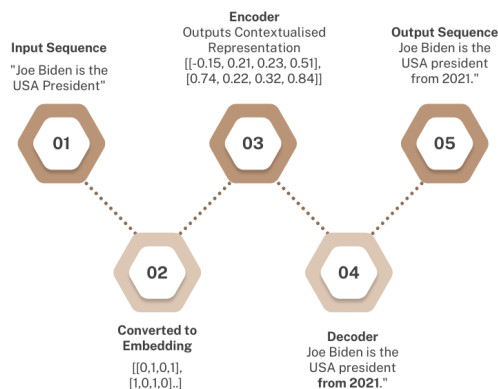


Figure 5: Transformer Architecture

An input sequence (i.e. sentence) passed into the transformer has its words first mapped to vector embeddings. Each word in the sentence now exists as a vector of numbers. This then goes into the encoder, which comprises multiple layers of attention networks that attempt to deduce the importance of each word relative to the entire sequence. This is where contextual information is captured, as multiple layers of feed-forward neural networks as well as attention layers are used to capture dependencies between words. The contextualised output is then passed to the decoder to generate the output sequence. The decoder will focus on the parts of the sequence that have been given the highest attention weights by the encoder, allowing it to choose output sequences with the highest probability. The transformer architecture is well suited for generative models, such as translation or textual generation. In our attempts to construct a fact-verification model, given an input sequence of a claim and evidence pair, we hypothesized that the attention mechanism in a transformer architecture will be helpful. For example, given a claim: “**Joe Biden** is the **president** of the **United States** from **2021**”, larger attention weights can be given to the key portions of the claim (i.e. “Joe Biden, president, United States, from 2021”) and lower weights to irrelevant information which do not affect the final fact labelling.

6.3 Transformers for Text Classification

Transformers are typically used for generative NLP problems, such as predicting the next word token given a sentence, paraphrasing sentences, or translating them. In order to adapt transformers for text-classification problems, a classification layer is added on top of the existing encoding layers. The layer would now output a vector of scores that represent the likelihood of a particular input belonging to a specific class. For easier interpretation, these scores are often converted to probability distributions. The class with the highest probability is then selected as the final label.

premise (string)	premise_binary_parse (string)	premise_parse (string)	hypothesis (string)	hypothesis_binary_parse (string)	hypothesis_parse (string)	genre (string)	label (class label)
"Conceptually cream skimming..."	"((Conceptually (cream skimming))) _"	"(ROOT (S (NP (JJ Conceptually) (NN cream) (NN skimming)) (VP (VBZ has)..."	"Product and geography are..."	"(((Product and) geography) ((are (..."	"(ROOT (S (NP (NN Product) (CC and) (NNL..."	"government"	1 (neutral)
"you know during the season and ..."	"(you ((know (during (((the..."	"(ROOT (S (NP (PRP you)) (VP (VBP know) (PP (IN during) (NP (NP (DT..."	"You lose the things to the..."	"(You ((((lose (the things)) (to (..."	"(ROOT (S (NP (PRP You)) (VP (VBP lose) (NP (DT..."	"telephone"	0 (entailment)
"One of our number will..."	"((One (of (our number))) ((..."	"(ROOT (S (NP (NP (CD One)) (PP (IN of) (NP (PRP\$ our) (NN number)))) (V..."	"A member of my team will..."	"(((A member) (of (my team))) ((..."	"(ROOT (S (NP (NP (DT A) (NN member)) (PP (IN of..."	"fiction"	0 (entailment)

Figure 6: MNLI Dataset (Premise, Hypothesis, Label)

Pre-trained transformers are available, and have been trained on significantly huge datasets. An example is the Multi-Genre Natural Language Inference (MNLI) task, which is a “crowd-sourced collection of 433k sentence pairs annotated with textual entailment

information” (Williams, Nangia, & Bowman, 2018). The transformer model predicts either the "entailment," "contradiction," or "neutral" class label given a premise and a hypothesis. Additionally, these transformers are based on the BERT architecture, which is bi-directional in nature. This means that given an input sentence, it is processed in both directions (left-to-right and right-to-left). This is especially important for contextual understanding, as it gives the model longer-range memory due to its ability to take into consideration context given both directions of the input sequence. To understand why bidirectionality is important, we illustrate an example: "The bank cannot function anymore because it's in financial distress." In this sentence, the word “bank” can take on separate meanings. For one, it can refer to a financial institution. Another definition is to refer to it as a bank of water. A unidirectional model will likely be only able to consider the word “bank” based on the words that have come before it (i.e. “the”). It would not have been able to take into consideration the words “in financial distress” that would make it clear that “bank” here refers to a financial institution. On the other hand, a bidirectional model like BERT will be able to consider both words that come before “bank”, as well as words that come after. This gives it better contextual understanding capabilities.

In the subsequent sections below, we detail our own experiments with transformer models on our fact classification task, and outline our results.

7. Model Implementation and Results

7.1 Dataset

Our dataset consists of 49627 claim and evidence pairs that are extracted from the FEVER dataset. The dataset contains multiple records, each containing information about a claim, its supporting evidence, and its label. Each record in the file has the following fields:

- **"id"**: A unique identifier for the claim.
- **"claim"**: The claim to be fact-checked.
- **"evidence"**: A list of evidence that is used to support or refute the claim. Each evidence is a list containing two elements: the title of the Wikipedia article where the evidence was extracted from, and the evidence itself.
- **"label"**: The overall label assigned to the claim by a human annotator. The possible label values are **"SUPPORTS"**, **"REFUTES"**, and **"NOT ENOUGH INFO"**, indicating whether the evidence supports the claim, refutes it, or there is

not enough to make a determination, respectively. The overall label could be decided based on multiple evidence sources.

- **"golden_evi"**: A list of labels corresponding to each evidence. They have the same possible values as the "label" field, but correspond to each claim evidence pair.
- **"claim_temporal_arguments"**: A list of temporal arguments extracted from the claim, such as dates, times or durations.

7.2 Preprocessing

We identified data issues with our original dataset. Some claim-evidence pairs had additional spaces in front of punctuation, or non alphanumeric characters. In order for the model to gain useful contextual references from our claim-evidence pairs, we preprocessed the claim and evidence pairs in a balanced manner. Too much preprocessing (i.e. removal of stopwords, stemmatizing) could alter the grammatical correctness or even meaning of the sentence, affecting prediction accuracy. We carry out the following preprocessing steps in order:

- Removing duplicate whitespaces
- Removing non-alphanumeric characters except for whitespace and normal punctuation
- Removing consecutive punctuations
- Removing leading and trailing punctuation except for normal end-of-sentence punctuation
- Removing spaces before commas and full stops
- Fixing possessive apostrophes
- Removing unnecessary quotes
- Removing duplicate whitespaces again

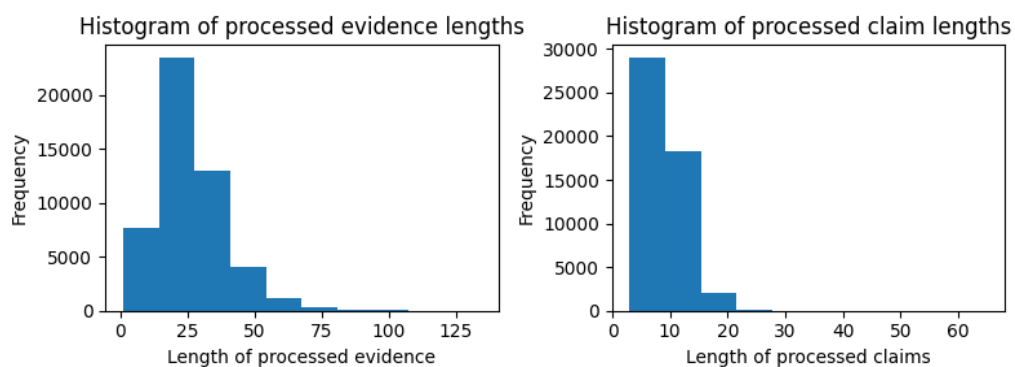


Figure 7: Histograms showing claim and evidence lengths

In our preprocessed dataset, the majority of evidence sentences are around 25 words, which is a reasonable length. A minority of evidence sentences are extremely long (i.e. 75 words). For claims, they are shorter than evidence sentences, and the majority of them are 8-10 words. This would make sense, since evidence sentences are extracted from Wikipedia articles, and likely to contain additional irrelevant content. For claims that are long, they likely contain multiple sub-claims, and these can be decomposed into smaller claims through an external decomposition module.

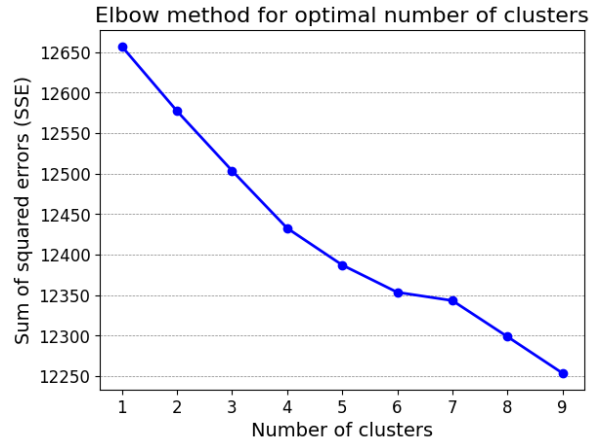


Figure 8: K-Means clustering for the claims

Additionally, to briefly visualise the diversity of different topics in our dataset, we used the elbow method to identify the optimal number of K-Means clusters for the claims. Here, we can see the elbow appears around the 6-7 cluster mark, and this is an indication that claims fall in a diversity of cluster topics. Briefly inspecting the dataset indicates that the claim domains range from music, politics, movies, history, covering quite a wide variety of topics.

In the 49627 claim evidence pairs, we sample a smaller subset of them to begin our model construction and evaluation. Since there were only 4290 pairs with the “REFUTES” label, we sampled a total of $4290 * 3$ (i.e. 12870) claim and evidence pairs for use as our final dataset to ensure that model training time is not unreasonably long and that final label proportions are kept even between the three different label classes.

7.3 Transformers

Using the [HuggingFace](#) model repository, we identified three different pretrained BERT based transformer models trained on an MNLI dataset. These models were selected as they obtained the highest accuracy scores on existing test datasets. The MNLI premise hypothesis task is relatively similar to claim-evidence verification, with the only difference being that

premises do not have to be factual in nature. We outline the three models selected in detail below.

Model Name	Model Description
roberta-large-mnli model	<ul style="list-style-type: none"> - By Facebook - Trained on Wikipedia dumps, Book Corpuses and also the MNLI dataset - Trained on Multi-Genre Natural Language Inference Dataset
deberta-v2-xxl arge-mnli	<ul style="list-style-type: none"> - By Microsoft Research - Improves the performance of Roberta models by using disentangled attention
albert-xlarge-vitaminc-mnli	<ul style="list-style-type: none"> - By Google Research and Toyota Technological Institute in Chicago - Aims to improve upon BERT architectures by allowing information sharing across different layers

Table 2: Different BERT based transformer models used

As an initial experiment, we tested the pre-trained transformers with our dataset extracted from FEVER (Thorne, Vlachos, Christodoulopoulos, & Mittal, 2018), with 12870 claim evidence pairs previously annotated.

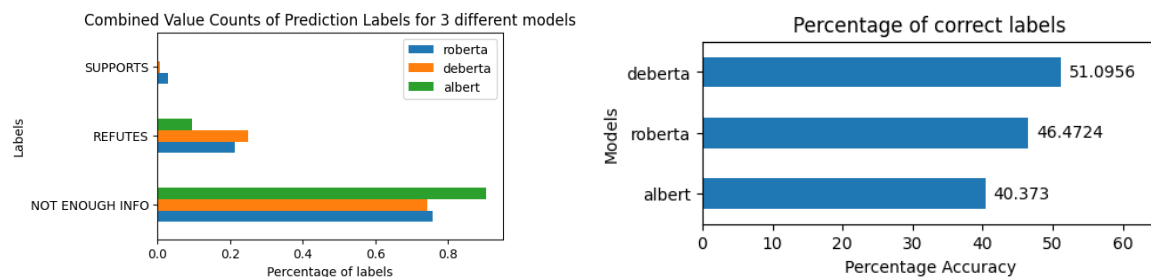


Figure 9: Prediction Accuracy using solely BERT based transformer models

From the results in figure 8, the pre-trained BERT based transformer models do not perform well when applied to our dataset. For all three models, they overwhelmingly predict the “NOT ENOUGH INFO” and “REFUTES” labels, suggesting a level of conservativeness in the transformer models. The Albert model did not predict a single “SUPPORTS” label. In terms of overall accuracy, the Deberta v2 model was most accurate, predicting labels correctly 51% of the time, with the Roberta and Albert models trailing behind at 46% and 40% respectively.

We further inspected some of the outputs of the transformer models to understand the limitations of such transformer models.

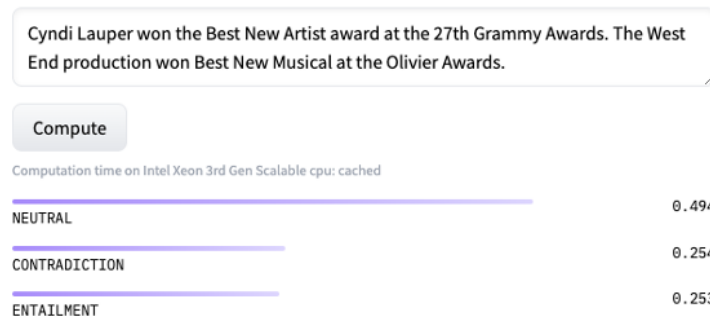


Figure 10: Output from Roberta

We found that transformer models do well when the identities of key entities being described is different. For example, in figure 9, “Cyndi Lauper” differs from “The West End Production”. The model recognises that these two phrases come at the start of their respective sentences, and the start of every sentence is typically more important. Hence, it outputs a “neutral” label, indicating its awareness that the evidence is likely irrelevant to address the claim.

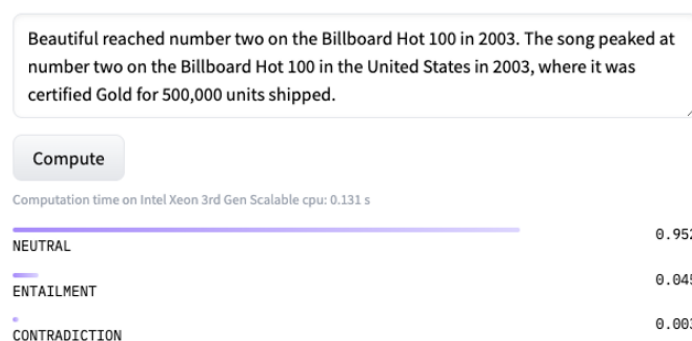


Figure 11: Output from Roberta

Here, we can again see that even though both sentences contain the “number two on the Billboard Hot 100 in the United States in 2003”, the second sentence contains a coreference “the song”, which likely refers to the song “Beautiful” in the first sentence. The model recognises that the claim specifically references the song “Beautiful”, but the reference in the evidence here maps to a generic song, and decides that there is not enough information for this evidence to support the claim, since the identity of this key entity is different.

There are other shortcomings with BERT models. For example, when the evidence is too long, or contains significant irrelevant information, the irrelevant information distorts the outcome since the transformer processes the entire input sequence. Here, the model is unable

to flag the contradiction that the film “Peggy Sue Got Married” is American, rather than Egyptian, instead returning neutral.



Figure 12: Output from Roberta

If the irrelevant parts of the evidence are removed, then the model gives us the correct label, which is a contradiction.

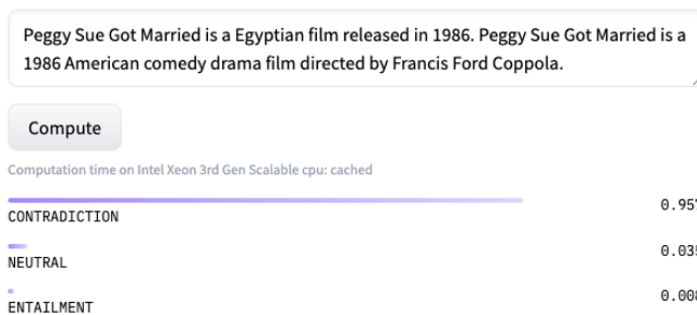


Figure 13: Output from Roberta

It appears that transformer models are sensitive with irrelevant information or noise. This means that we cannot use transformers as standalone predictive mechanisms in our model to make fact-verification decisions. However, they perform better when deciding contradiction labels, with the Deberta V2 model achieving a decent accuracy of 51% above the baseline of 33% (since all labels were evenly distributed). Hence, we decided to include its probability outputs as an initial feature in our own fact verification model.

7.4 Additional Feature Extraction

In order to further improve the accuracy of our predictive model, we had to identify additional key features that could affect the truth values of a claim-evidence pair. We considered features that were as general as possible to any claim and evidence pair, since selecting overly specific features could result in overfitting during the testing phase.

First, we extracted entities and noun chunks in the claim and evidence pair using the Spacy package. Entities could also be named or unnamed, and named entities refer to specific objects, people, places or concepts. Normal entities are more general (i.e. “the cat”, “the man”). Noun chunks, on the other hand, refer to general items as well, but refer to groups of words that are almost always seen together (i.e. “The brown cat” is a noun chunk in the sentence “the brown cat walked into my house”). If we only extracted named entities, then we would miss out on claims which do not refer to specific entities, but only general occurrences. (i.e. “a plane crashed today in the USA”). Noun chunks give us the added ability of extracting the words that would most probably surround an entity, capturing the grammatical structure of the entities of the claim and evidence pair.

Additionally, we hypothesized earlier that a shortcoming with the BERT transformer model was the lack of coreference resolution. The models did not know if “she” in the evidence sentence was likely to be referring to the same individual in the claim. To address this, we use the “fastcoref” Python package, which returns us a list of coreferences in both the claim and evidence. This then gave us the means to flag if there were corefering entities in the evidence. Knowing the semantic similarity between the claim and evidence pair could also be useful. If the evidence is semantically more similar to the claim, then it is also likely to be more relevant. Since temporal elements of the claim and evidence can also affect annotation accuracy based on our earlier experiment on the FEVER dataset, we extracted them separately using a temporal tagger built on the BERT infrastructure. Even if the general facts of the evidence support the claim, a different time or date would render the support invalid.

On top of all these features, we also extracted numbers or quantities from both the claim and evidence, as well as calculated the counts of entities, noun chunks and quantities extracted. One last challenge we had was to generate similarity scores between the named entities, normal entities, noun chunks and temporal information extracted., because most models can only take in numerical features as inputs, while our entity information was returned in a list of strings. This was challenging and we had to deal with the following situation:

Claim: Joe Biden is the president of the United States.

Evidence: Joe Biden assumed presidency of the USA.

Named entities returned for claim: [['Joe Biden'], ['United States']]

Named entities returned for evidence: [['Joe Biden'], ['USA']]

The objective is to get a similarity measure of the extracted named entities between the claim and the evidence. If the match was done by only comparing string similarity, then “Joe Biden” will match in both claim and evidence, but “United States” will not match with “USA”. If an attempt was instead made to compare only contextual similarity, then some terms, such as specific names like “Joe Biden” might not even be present in the word embeddings used (i.e. GloVe). In order to consider both semantic and string similarity, the following formula was used to calculate the similarity between the entities returned:

$$\text{Combined Similarity} = \alpha * \text{Embedding Similarity} + (1 - \alpha) * \text{Levenshtein Similarity}$$

where:

- **α** : the weight or importance given to the embedding similarity
- **Embedding Similarity**: the cosine similarity between the word embeddings of two words (contextual similarity)
- **Levenshtein Similarity**: the normalized Levenshtein distance between two phrases (string similarity)

We used an α of 0.5 in our experiments, but this can be further fine-tuned. The combined similarity metric was averaged for each entity or item in a list of strings, and considers both the string similarity and contextual similarities of the strings being compared.

Feature	Package Used
Bert based transformer probabilities	Huggingface Transformers
Named Entities, Normal Entities and Noun Chunks (similarities)	Spacy and Combined Similarity Function
Coreferences	fastcoref
Similarity score between claim and evidence	Huggingface’s all-MiniLM-L6-v2 sentence-transformer
Temporal Information (similarities)	Huggingface’s temporal_tagger_BERT_tokenclassifier and Combined Similarity Function

Numbers	Spacy
---------	-------

Table 3: Features extracted to be used in model construction

7.5 Classifier Model

With the features that we extracted, we moved on to the construction of our classifier model. A train-validation-test split of 8-1-1 was used to construct the training, validation and test dataset. Numerical features were scaled, and combined with another set of word embeddings of both the claims and evidence sentences generated using GloVe. Different model architectures were explored and tested below.

7.5.1 Decision Tree, Random Forest, Gradient Boosting

A decision tree was used initially.

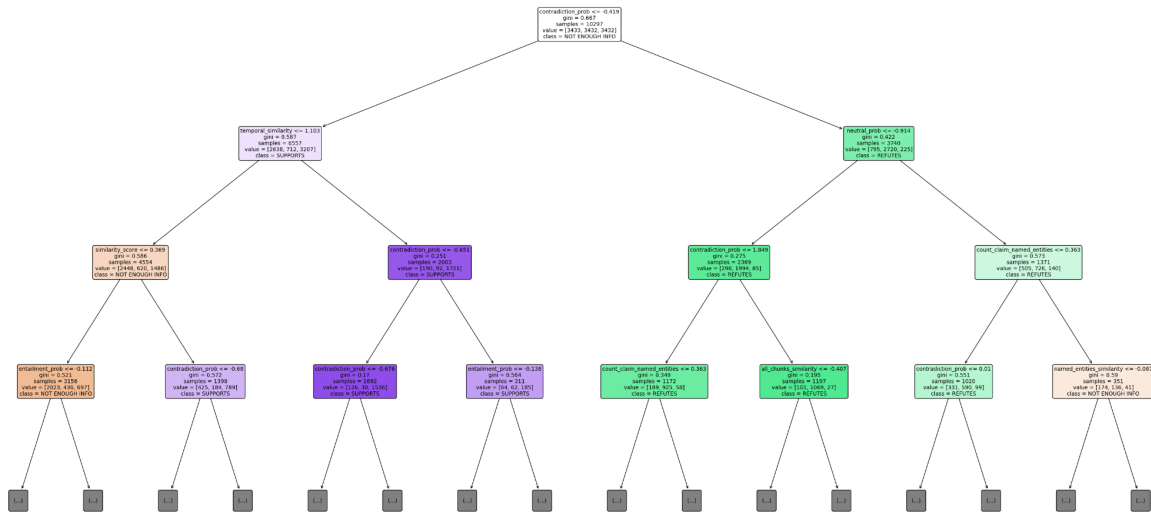


Figure 14: A segment of the decision tree plotted

The decision tree appears to support our earlier conclusions. Initially, all pairs are put into the “not enough information” category. The model looks at the contradiction probability features generated by the transformers. If there is a low contradiction probability, then pairs move to the “support” class, and the model now considers the temporal similarity of the claim evidence pair. If the temporal similarity falls below a threshold, the pairs go into the “not enough info” category. If there is a high contradiction probability, then it is likely that the claim-evidence pairs are either labelled “not enough information” or “refutes”, and the decision tree then considers the “neutral probability”. One downside of a decision tree is the

tendency to overfit, but visualisations of the tree enable the ability to understand how the tree arrived at its final predictions.

Classifier type	Class	Precision	Recall	F1-Score
Decision Tree (Overall test accuracy of 64%)	Not Enough Info	0.58	0.56	0.57
	Refutes	0.64	0.66	0.65
	Supports	0.70	0.70	0.70
	Weighted Avg.	0.64	0.64	0.64
Random Forest (Overall test accuracy of 74.5%)	Not Enough Info	0.69	0.68	0.68
	Refutes	0.76	0.78	0.77
	Supports	0.78	0.78	0.78
	Weighted Avg.	0.74	0.75	0.74
Gradient Boosting (Overall test accuracy of 78.6%)	Not Enough Info	0.76	0.72	0.74
	Refutes	0.79	0.83	0.81
	Supports	0.81	0.82	0.81
	Weighted Avg.	0.79	0.79	0.79

Table 4: Test results of tree based classifiers

Our results on the test set show that gradient boosting outperforms both random forests and decision trees. It aggregates multiple decision trees to deal with complex data and significant noise in the dataset, reducing the risk of overfitting to the train dataset. It also appears that the classifier does better on classifying the “Supports” label, as compared to the “Refutes” or the “Not enough info” label.

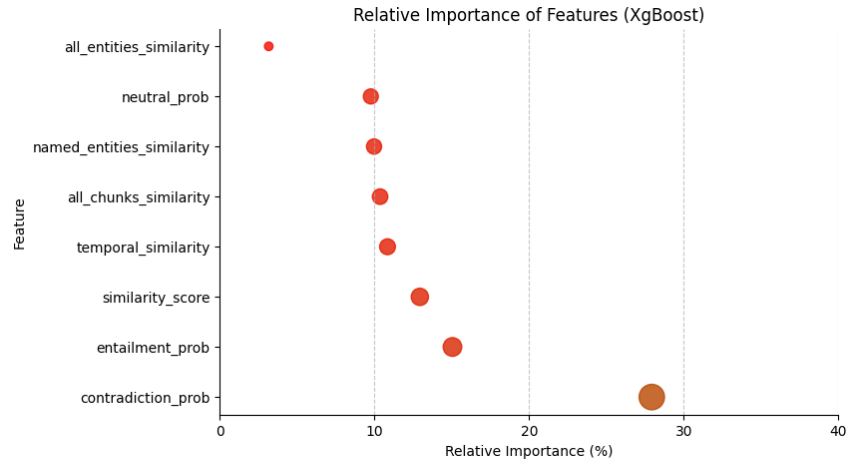


Figure 15: Feature Importance Plot

As multiple key features were extracted, some of them were likely not as useful for the classification process, and should be removed to further reduce noise in the overall dataset. Here, we rely on the feature importance plot from the gradient boosting classifier, which indicates that the probabilities generated by the BERT transformers, as well as similarity measures of the named entities, noun chunks, and temporal elements were most important. The counts and the coreference flag were likely not useful. Hence, we reduce the final feature set to the following:

Final Features
Contradiction, Neutral, Entailment Probability from BERT based transformers
Counts of claim and evidences' named entities
Similarity score between claim and evidence
Similarity score between normal and named entities in claim and evidence
Similarity score between noun chunks in claim and evidence
Similarity score between temporal elements in claim and evidence

Table 5: Final Features of the Gradient Boosting Model

7.5.2 Support Vector Machine

SVMs were also explored, as they are able to handle high-dimensional data well. As we used a 300 dimension GloVe embedding, SVMs could perhaps perform better at the classification task. They also perform well in unseen data situations, and avoid overfitting. Different kernel functions are also available to capture different relationships. In our SVM approach, we obtained a test accuracy of 76%, which was not as high as the gradient boosting classifier, and we similarly extracted the most important features.

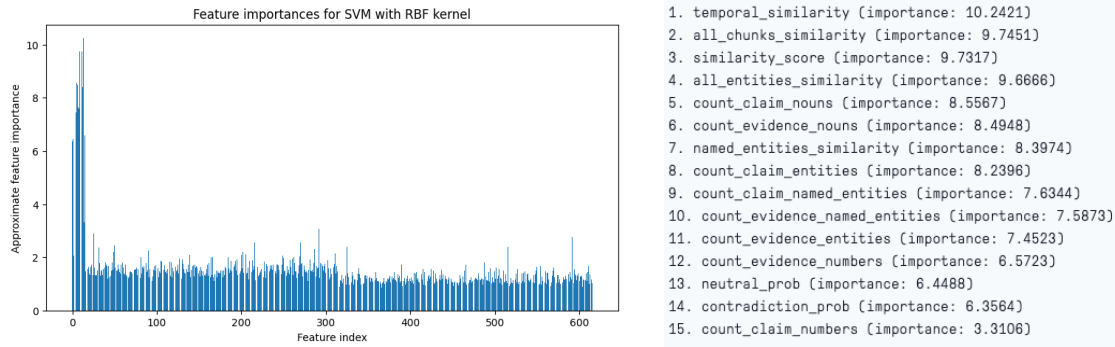


Figure 16: Approximate Feature Importances for SVM

For the SVM, the temporal, noun chunk and semantic similarity scores dominate in terms of their relative importance to the model. Interestingly, the SVM did not rely as much on the BERT transformer probabilities as the tree-based classifiers. We started out with a RBF kernel, the most commonly used and attempted to fine-tune the SVM, or include a subset of the original features, but received no further improvements.

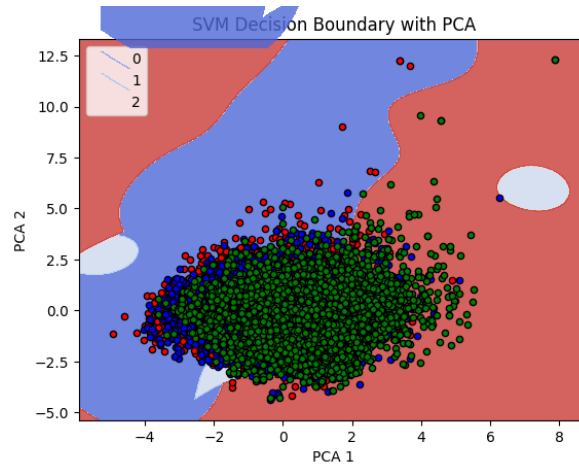


Figure 17: SVM Decision Boundaries with PCA

Though PCA was carried out to reduce all of the features to a 2D plot, it is clear that the decision boundary was not linear. There were also multiple overlapping data points, so it

would suggest that a non-linear SVM kernel might be better able at capturing the non-linear relationships. However, this did not give us significant improvements during the fine-tuning process.

7.5.3 Neural Network

Lastly, experiments were done on a LSTM (Long Short-Term Memory) model. The architecture used is briefly outlined below, with an additional attention layer added to better grasp the importance of different parts of the input sequences.

1. Input Layers

- Claim Input
- Evidence Input
- Feature Input

2. Embedding Layer

- Apply to Claim Input
- Apply to Evidence Input

3. Bidirectional LSTM Layer

- Apply to Claim Embedding
- Apply to Evidence Embedding

4. Attention Layer

- Apply to Claim LSTM output
- Apply to Evidence LSTM output

5. Merge Layer (Concatenate)

- Merge Claim LSTM output, Evidence LSTM output, and Attention weights

6. Feature Input Layer (Dense)

- Apply to Feature Input

7. Flatten Layers

- Flatten Merged LSTM and Attention Layer outputs
- Flatten Feature Input Layer output

8. Concatenate Flattened Layers

----- Concatenate Flattened Merged LSTM and Attention Layer outputs with Flattened Feature Input Layer output

9. Dense Layers

- Dense Layer 1 (ReLU activation)

----- Dense Layer 2 (Softmax activation)

Different combinations of LSTM parameters and model types were trialled and the results are reported below.

Model Type	Weighted Precision	Weighted Recall	Weighted F1
Bi-directional LSTM with dropout of 0.35 and L2 Regularization	0.76	0.75	0.75
Increased dropout to 0.4	0.75	0.75	0.75
Unidirectional LSTM, dropout of 0.4	0.75	0.75	0.75
Stacking multiple bi-directional LSTM layers	0.74	0.73	0.73
Using a different word embedding Fasttext by Facebook instead of GloVe	0.75	0.75	0.75
Changing batch size to 64	0.76	0.75	0.75
Adding additional dense layers	0.76	0.76	0.76
Using GRU architecture	0.75	0.75	0.75
Changing the attention weights to simple dot-product	0.75	0.75	0.75

Table 6: LSTM experiments and results

With the LSTM model, it was incredibly difficult to fine-tune. While matching the performance of the SVM, it performed worse than the gradient boosting model. There were also instances where adding too much complexity to the model resulted in overfitting. The training accuracy went up to over 80-90%, while test accuracy hovered around ~72%. This is another instance when a complex model does worse than a simpler one.

7.6 Generative Pre-Formed Transformers (GPT)

OpenAI’s publicly released product – GPT 3.5, which according to the BBC, is “capable of generating human-like text” (Hughes, 2023) was released midway into our research. Due to the powerful nature of GPT 3.5, and perhaps even more so its successor GPT 4.0, we found it important to explore the model’s ability and potential for use in our text-classification problem. It was demonstrated in a previous version of GPT (GPT 2.0) that it was effective in wide-ranging language tasks, from sentiment analysis, text generation and even question answering (Radford et al., 2018).

For GPT to perform specific tasks, a small amount of labelled training data can lead to a significant increase in accuracy (i.e. 10 percentage points in sentiment analysis with Yelp’s sentiment analysis dataset) (Hosseini-Asl, Liu, & Xiong, 2022). This points towards the huge potential of GPT to be deployed in a classifier model, despite its most common use in text generation research. We were however initially sceptical about GPT performance with minimal training data. As with transformer architectures, they typically require significant training data. For example, GPT 3.0 was trained with 45 terabytes of textual data. With smaller quantities of training data, models like GPT might be especially sensitive to perturbations within the training data.

7.6.1 Prompt-Based Tuning

In our initial trial, we first ran the prompt-based GPT 3.5 Turbo model hosted by OpenAI via our own Google Cloud Function (<https://fyp-gpt-wr6verfzva-as.a.run.app>) to evaluate its performance in predicting General and Temporal labels on the same test-set of 1287 claim and evidence pairs that was used to generate test statistics for our final XGBoost model. Using a prompt-based model means that our prompt would have to be significantly fine-tuned. This was challenging, and we had to find a structured way to approach the prompt-tuning process. Inspired by research carried out by Shinn, Labash, & Gopinath (2023), we noted that autonomous agents benefit from being tasked to reflect on their actions. Instead of driving the prompt engineering process ourselves, we drove revisions by a series of reflection learning processes within the GPT 3.5 model. In our prompt-tuning with GPT 3.5, we experimented with an iterative process incorporating reflection learning, and we outline an example below:

Step 1: Initial Prediction

Claim: System of a Down briefly disbanded in limbo.

Evidence: System of a Down is the debut studio album by Armenian American metal band System of a Down, released on June 30, 1998, by American Recordings and Columbia Records.

GPT 3.5's initial prediction:

Temporal Label: SUPPORTS

Justification: The evidence states that the debut album was released on June 30, 1998, which matches the time frame mentioned in the claim.

Clearly, GPT has gotten it wrong, but we want to understand why the model believed that the time frame was mentioned in the claim.

Step 2: Understanding what went wrong

Then, we get the model to explain how it came to its decision.

Why did the model make the decision of "Supports" for the temporal label?

Based on the evidence provided, the model might have determined that the date mentioned in the evidence (June 30, 1998) matches the time frame mentioned in the claim ("briefly disbanded in limbo"). As a result, the model may have concluded that the evidence supports the temporal aspect of the claim. However, this conclusion is incorrect as the date of the debut album does not have any relevance to the claim.

Step 3: Correcting itself

We then task the model to correct itself, since the reflection has triggered an awareness of the incorrect conclusion.

Indeed, so if the model is corrected, what should it output?

If the model is corrected, it should output the following:

- TIME_LABEL: "NOT ENOUGH INFO"
- TIME_JUSTIFICATION: "The evidence provided does not contain any information related to the temporal aspect of the claim, and therefore there is not enough information to support or refute the temporal aspect of the claim."

This output correctly reflects that the evidence provided does not address the temporal aspect of the claim and there is not enough information to support or refute it.

Step 4: Refining our prompt

Lastly, we then ask the model how we can correct the prompt supplied to reduce the odds of the mistake recurring. In this case, we simply add in the phrase “...Ignore irrelevant elements of the evidence when making judgments about the temporal aspects of the claim and evidence.”...

Table 7: Prompt-tuning process through reflection-learning

The final prompt used for GPT 3.5 is appended in the appendix. With the self-reflection in place, coupled with the prompt-tuning, we observe the following improvements to the fact checking accuracy when GPT is tasked to predict the same set of claims.

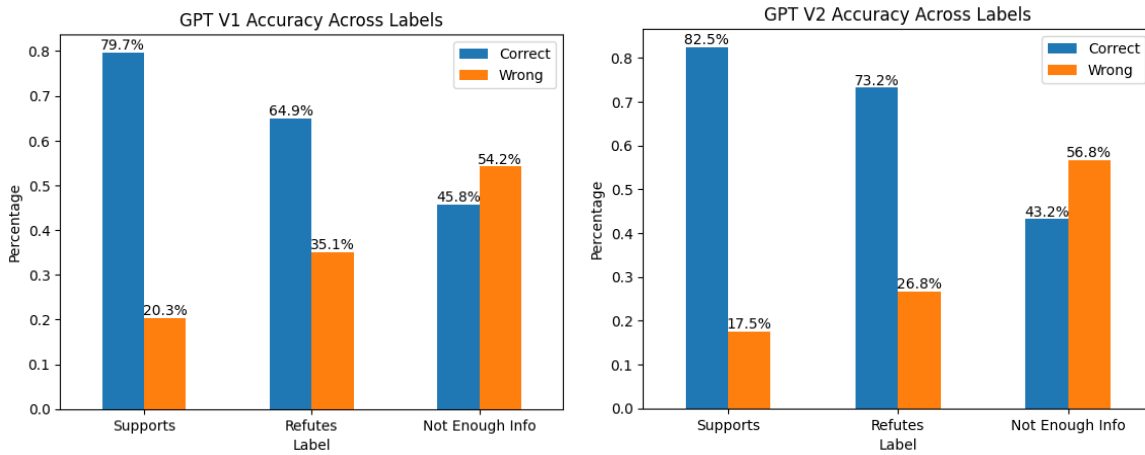


Figure 18: GPT 3.5 accuracy in fact checking before (V1) and after (V2) prompt tuning

Here, GPT V2 represents the prompt-engineered model (after several rounds of prompt tuning). Overall accuracy increased from **63% to 66%**. The GPT model became slightly better at predicting the supports and refutes label, while suffering from a slight fall in the accuracy of “not enough info” labels. This suggests that prompt tuning driven by reflection learning could be a quick and viable way to fine-tune the GPT model.

However, this also means that GPT 3.5 is especially sensitive to nuances in the prompt supplied. One has to be clear about the requirements, and also define terminologies that can be interpreted differently. Another downside of such prompt engineering was that the model’s input would be extremely long and convoluted, and it would be impossible to exhaustively define every single situation.

7.6.2 Non Prompt-Based tuning

In order to address the deficiencies of a prompt-based tuning approach, we decided to explore the possibility of fine-tuning a separate GPT 3 model that would no longer require a long and convoluted prompt to generate results. We make use of the baseline ada model provided by OpenAI. At this stage, we were inspired to test two different versions of fine-tuning. The first version involves simply feeding 8476 claim-evidence pairs randomly extracted from our training dataset used for the other classifier models into the GPT 3 model together with their annotated class. The second version involves combining the features that we extracted in the earlier part of our research, and feeding them together with the claim-evidence pairs. This would effectively allow GPT to take into consideration the outputs of both the earlier transformer models, as well as the additional features we extracted, leveraging on the strengths of different approaches.

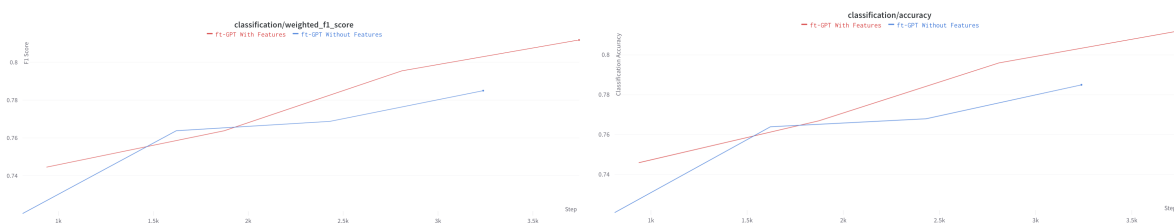


Figure 19: Classification F1 Score and Accuracy of Fine-tuned GPT models with and without additional features

The results from figure 10 indicate that with fine-tuning, both models perform exceptionally well, with the GPT model with additional features added (red line) doing slightly better at prediction accuracy compared to the model without the additional features (blue line). The GPT 3 model without features recorded a classification accuracy of 78.5%, while the one with features recorded an accuracy of 81.2%. These models will be further evaluated in the final model evaluation section below.

7.7 Final Model Evaluation and Selection

With our model construction complete, we proceed to evaluate the three best models identified (GPT 3.5 with prompt-tuning, fine-tuned GPT 3 with features, Gradient Boost) on a separately constructed fixed test set retrieved from the FEVER dataset (Thorne et al., 2018) that has not been seen by any of the models during training or validation. This means that the test set is not only unique in terms of its claim and evidence pair, but also contains both unique claims, as well as unique evidence sentences, and would be a more accurate reflection

of the model’s capabilities at unknown data. We used the XGBoost implementation of Gradient Boosting in our test runs.

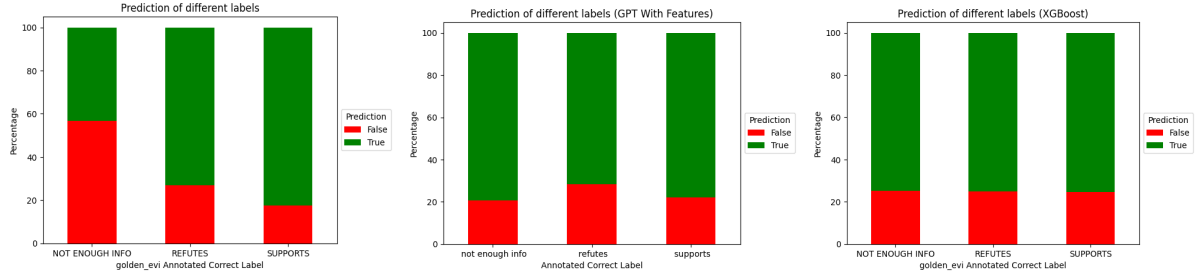


Figure 20: Accuracy of GPT 3.5 vs Fine-Tuned GPT 3 and XGBoost

We find that the overall accuracy of the prompt-tuned GPT 3.5 in predicting the correct labels, based on annotations done, is 66%, compared to ~75% for our XGBoost model and ~76.2% for the fine-tuned GPT 3 model with features added. The fine-tuned hybrid GPT 3 model with additional features added outperforms both the prompt-tuned GPT 3.5 model and the XGBoost classifier. When we split this up by the different labels, we realize that both GPT 3.5 and the fine-tuned GPT 3 do especially well at predicting labels that are “SUPPORTS”, with about 80% accuracy, which is marginally better than the XGBoost model. The prompt-tuned GPT 3.5 model does especially badly at predicting “NOT ENOUGH INFO” labels, suggesting again its sensitivity to the prompt used. GPT models also retain other benefits. For one, the prompt-based GPT 3.5 model is more explainable. Majority of neural networks or gradient boosting trees experience the significant downside of being “black-boxes”, where predictions could be accurate and precise, but there was no way to know entirely how the model arrived at its final decision through the complex network of layers and attention weights. However, for the chat-based GPT 3.5, being a text-generation model, it is possible for us to add labels to request the model to justify its own predictions.

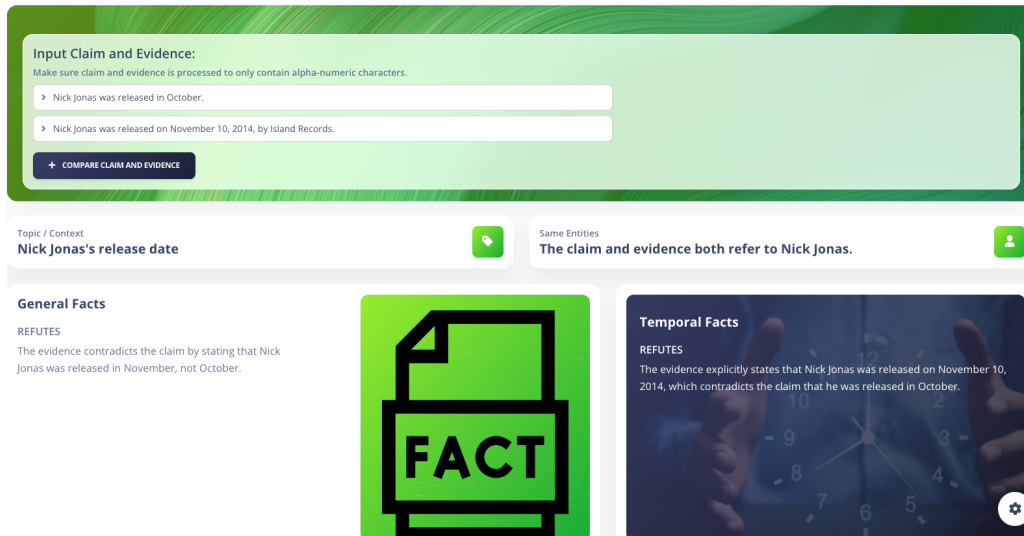


Figure 21: An example dashboard showing how GPT provides more explainability

For deployment on our web-based annotation platform, explainability was likely a key consideration, since it would benefit investigators in the manual review phase, where they have to compare the model’s predictions with the annotators’ labels. The second key consideration was the ease of deployment. GPT models are easily deployable for inference via REST-APIs, whereas our XGBoost model will be difficult to deploy for real-time inference, due to the need to find separate GPU hosting services. Hence, though the fine-tuned GPT 3 model produced the most accurate classification results, we eventually decided to deploy the prompt-tuned GPT 3.5 model on our web platform, as it provides reasonable accuracy (~66%), and is most importantly, explainable. The more accurate GPT 3 model can however, be continuously fine-tuned with annotation results from the annotation platform. With model development and selection complete, we move on to discussion on our web-based annotation platform, which forms the software engineering part of the project.

8. Web-based annotation platform

The web-based annotation platform was designed to demonstrate how predictive technologies can benefit the process of annotation. It incorporates both the predictive model that we developed earlier, as well as features that would enable the annotation of temporal claims in a swifter and more efficient manner. A video demonstration of the web platform can be viewed here: https://www.youtube.com/watch?v=T_aWgKGgc9M

8.1 Design Decisions

8.1.1 Overall Pipeline

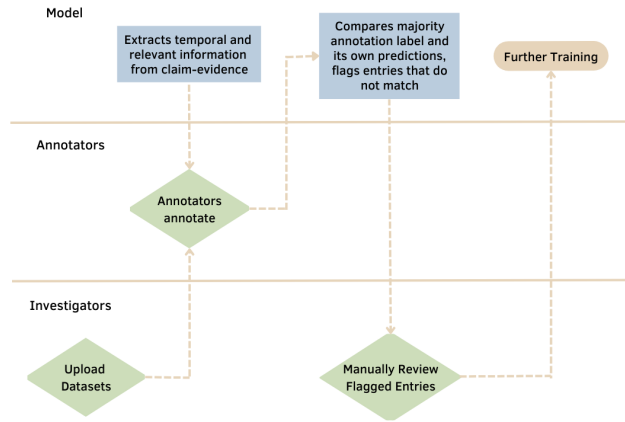


Figure 22: Overall Pipeline of Annotation System

The pipeline was designed with simplicity in mind. The predictive model will aid both the annotator and the investigator. For the annotator, GPT 3.5 is able to extract temporal elements of claim and evidence sources and we highlight these portions during the temporal label annotation. In a long evidence sentence of an average of 25 words, it might sometimes be difficult and also time-consuming for annotators to manually extract the temporal elements, and make comparisons. Doing such pre-annotations has the potential to result in significant time savings of 13-20% per annotation, with no statistically significant deviations in annotation performance (Lingren et al., 2014). The model also predicts its own general and temporal labels for the claim-evidence pair, and compares the majority annotation label with its own predictions. If they do not tally, then the claim-evidence pair is sent for manual review. Here, we considered if we should show the model’s predictions to the annotator as well, to aid the annotator in making his or her annotation decisions. We decided against it, as this has the potential to bias the annotator to simply annotating based on the model’s outputs. If annotators are able to view the model’s predictions, then it would also not be possible to assume that annotators’ accuracy is independent of that of the model’s accuracy. With the addition of the model’s predictions to sieving out manual reviews, this reduces the investigator’s workload, and also ensures that only annotation entries with the highest probabilities of being wrong are manually reviewed.

8.1.2 Database

We needed a database to store our user information, as well as datasets for annotation. We considered two different approaches. The first made use of NoSQL databases such as MongoDB, which are more proficient in storing unstructured data. The second was to go with MySQL databases. We went with the MySQL approach as fact-verification datasets are often structured (i.e. with claim, evidence pairs), and claim and evidence pairs were often linked to each other. The relational nature of a MySQL database will ensure that data access, as well as organisation is more efficient. MYSQL databases can also handle bulk data volumes (60000 annotation entries in a single database) well with bulk queries. Seven key tables are defined. CustomUser stores administrator information and the other tables store both the claim and evidence information, as well as annotations done by the annotators. The MYSQL backend database communicates with the Vue frontend through REST API calls.

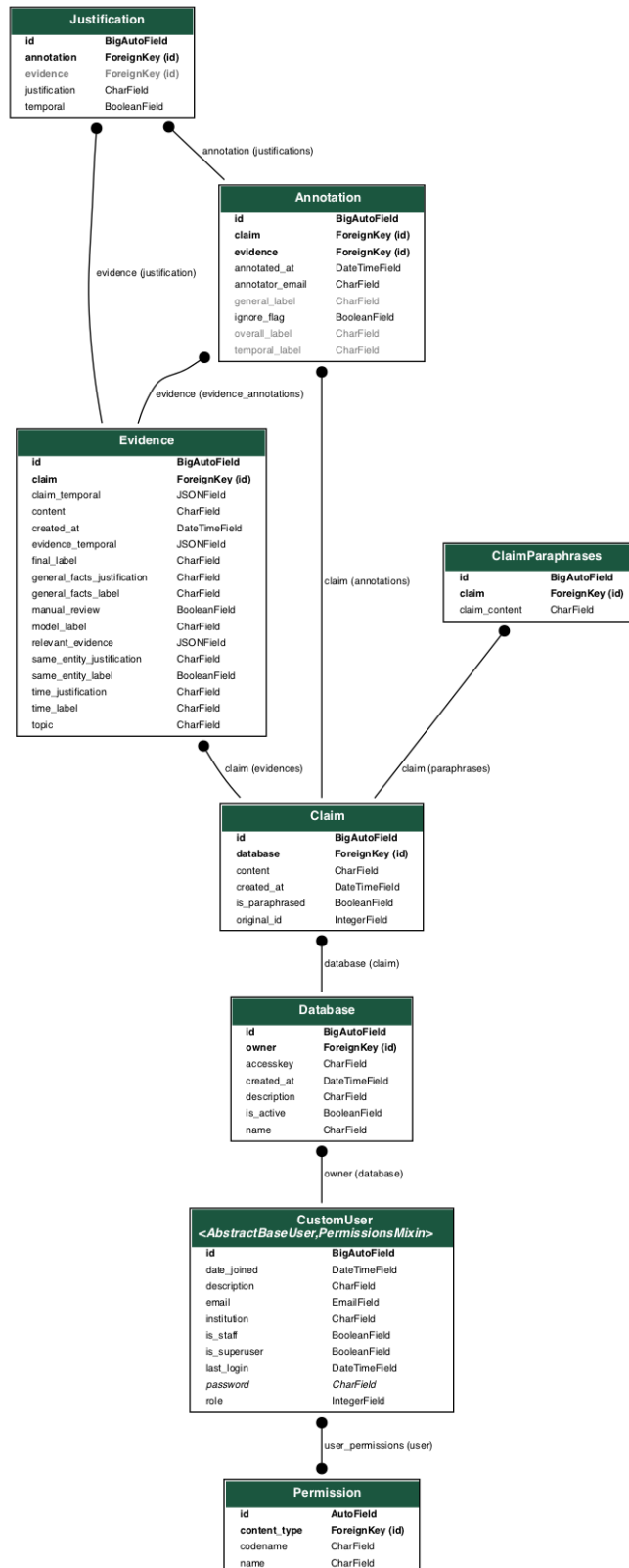


Figure 23 - Database Diagram

8.1.3 Backend Infrastructure

When designing the backend, we had multiple options. Our first consideration was to look for a Python based backend, as our GPT predictive models were also written in Python. Language compatibility would allow us to seamlessly call the predictive models from our backend server. This immediately singled out Django and Flask, which are two of the most popular backend languages. Django also provided the additional benefit of being packaged with a SQLite database, which increased the ease of development. Additionally, there were built-in security and authentication features that gave us the ability to focus on the key features of the project. Most importantly, it also supported the REST-API framework to communicate with our front-end platform.

8.1.4 Frontend Infrastructure

As for the frontend, we narrowed it down to two different languages that we were familiar with - Vue and React. We eventually chose Vue as it was much simpler and more intuitive than React, and it also provides a two-way data binding system, whereby changes in data variables will automatically trigger an update in the user's view. This allows us to manage state changes in a much more efficient manner.

8.2 Overall Architecture

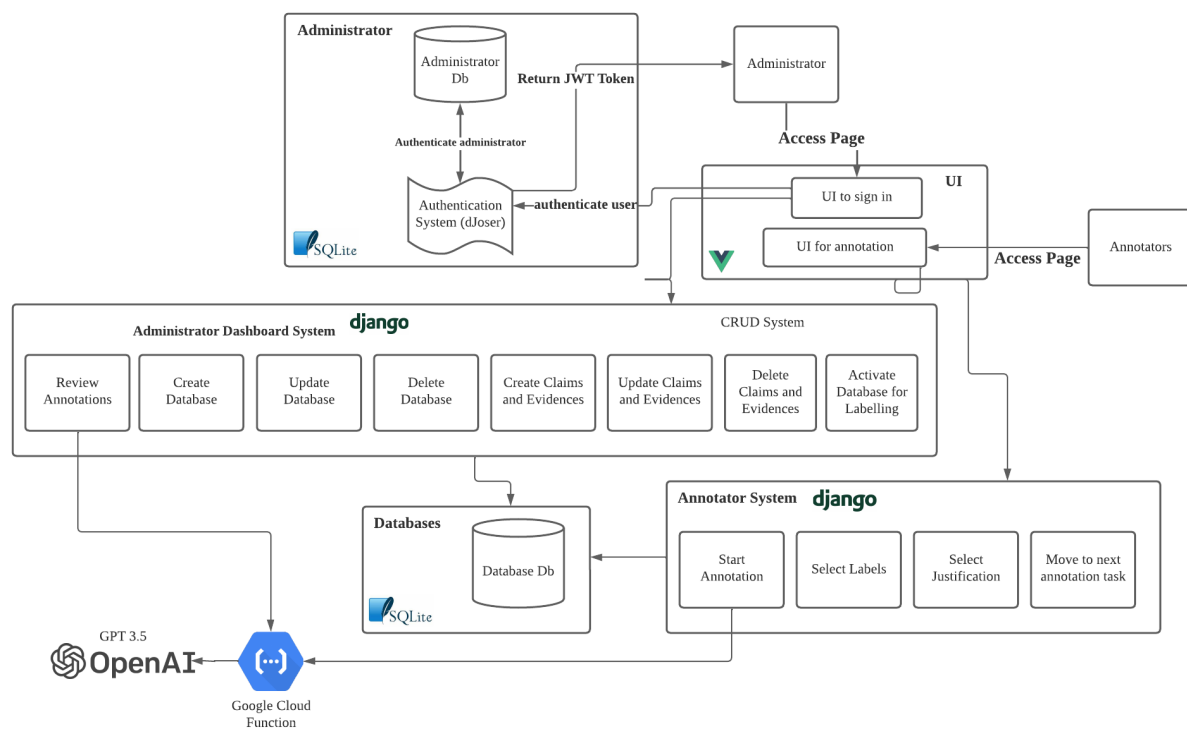


Figure 24 - Overall Architecture Diagram

There are two main roles in the annotation system - annotators and the investigators. Both user groups access the same user-interface to sign-in, with the investigators requiring basic authentication. Annotators have their own annotation page, which sends API calls to a backend Google Cloud Function that retrieves model information from OpenAI's GPT models. Predictive model information is cached in the SQL database, and there is only a need to generate it once. Detailed user flows are illustrated in the Appendix section.

8.3 Tech Stack

Component	Technology Used
Front-end	Vue-3 with Bootstrap for styling
Local State Storage	Pinia
Back-end	Django, Google Cloud Functions
Storage	Native Django SQLite Database
Predictive model	OpenAI's GPT 3.5

Table 8 - Tech Stack

8.4 Evaluation

In order to evaluate the effectiveness of our annotation platform in enhancing annotation quality, we engaged a third party annotator from Fiverr, and uploaded 100 random claims and evidence sentences for the annotator to annotate. Clear and concise annotation requirements were given to the annotator via a Google document, and the annotator was also compensated for his efforts. The annotator was told to try his best in annotating as accurately as possible, which is similar to the advice given at other annotation settings. After the annotator completed 100 annotations on our platform, we then manually accessed the annotation for accuracy. Out of the 100 annotations done, 92% were accurate. In our earlier analysis of the annotated FEVER dataset, we noted that even after manual review, its annotation accuracy is only 77.1%. Without having gone through to the manual review phase, our user test has already achieved a 92% annotation accuracy, outperforming conventional annotation tools in terms of annotation accuracy. We assume that all other factors are constant, and that annotators would typically perform their duties to the best of their abilities. A larger scale user study is necessary to further confirm this finding, but we are unable to do so due to resource limitations.

We also would like to point out the cost-effectiveness of our annotation system. Each invocation of the predictive model costs approximately \$0.002 USD - \$0.004USD, but delivers a marked improvement in overall annotation quality, as well as reduced review workloads for the investigator.

9. Conclusions

9.1 Summary

In our research, we have explored multiple transformer models for fact-checking tasks, and explored the newly released GPT transformer’s capability to fact check. Based on the model research, we have also deployed GPT’s capabilities in an online annotation platform system to support the development of a temporal fact-verification dataset and ensure high annotation quality. Evaluation of our models show that the best-performing models record close to 76% accuracy on an unseen claim-evidence test dataset, which is almost close to the annotation accuracy that we calculated based on the FEVER dataset. Evaluation of our web-based annotation platform also indicates that there is potential for the system to improve annotation quality simply from the pre-annotations that the predictive model provides to the annotator. With the pre-annotations, annotation accuracy on a small-scale user test was recorded as 92%, surpassing the 77% of the FEVER dataset, which was annotated without any predictive model support. All project files can be accessed here: <https://github.com/yejiadong/fyp>

9.2 Limitations

Computational resource availability was the main limiting factor in our research. While we started out ambitious, we realised that the transformer models required significant GPU power to train and finetune. With only an NVIDIA A4000 GPU, each model would require at least 4 hours to train and fine-tune. Resultantly, we had to scale back on the training data used, and this might have affected accuracy. This is why an annotation platform is critical for not simply the evaluation of classifiers, but also to further fine-tune and improve their accuracies through the generation of accurately annotated data entries. Also, we want to point out that our user study for our annotation platform was extremely small-scale due to the costs of running such a study. A single annotator was co-opted, and his views and results were studied. A larger scale study incorporating annotators from different backgrounds, as well as actual investigators would have delivered much more reliable results.

9.3 Recommendations for Future Work

There is still potential room for further study. To further improve the fact checking classifier’s accuracy, we can continue to explore further combinations of various model outputs, or even pass final predictions from different models through a voting mechanism to exploit the strengths of different models – some being slightly “cautious”, and others relatively versatile with evidence sentences.

On the developmental front, it is possible to also feed the reviewed annotation entries automatically back into any deployed fact-verification models for an active learning process to kick in. This allows the deployed model to learn from its mistakes, and gradually get better at the specific task of fact verification. It would be even better if this model was the same one that made the original inaccurate prediction, as conveying error information to the model kick-starts a process of reflection learning where the model realizes where and why it has predicted the label wrongly. Transformers require a large amount of training data, so the annotation platform can also be integrated directly with the training of fact verification models to create a self-sustaining process where the model consistently learns from reviewed annotations.

References

- Atkinson, C. (2019, April 26). *Fake news can cause 'irreversible damage' to companies - and sink their stock price*. NBCNews. Retrieved November 1, 2022, from <https://www.nbcnews.com/business/business-news/fake-news-can-cause-irreversible-damage-companies-sink-their-stock-n995436>
- Atske, S. (2021, September 20). *News consumption across social media in 2021*. Pew Research Center's Journalism Project. Retrieved November 1, 2022, from <https://www.pewresearch.org/journalism/2021/09/20/news-consumption-across-social-media-in-2021/>
- Center for Information Technology and Society - UC Santa Barbara. (n.d.). *The danger of fake news in inflaming or suppressing social conflict*. Retrieved November 1, 2022, from <https://www.cits.ucsb.edu/fake-news/danger-social>
- Chiusano, F. (2022, February 12). A brief timeline of NLP from Bag of Words to the Transformer Family. Medium. <https://medium.com/nlplanet/a-brief-timeline-of-nlp-from-bag-of-words-to-the-transformer-family-7caad8bbba56>
- Conner-Simons, A. (2021, March 29). Major ML datasets have tens of thousands of errors. MIT CSAIL News. Retrieved from <https://www.csail.mit.edu/news/major-ml-datasets-have-tens-thousands-errors>
- Flatow, D., & Penner, D. (2017). On the robustness of convnets to training on noisy labels.
- Hahn, U., Beisswanger, E., Buyko, E., & Faessler, E. (2012). Active Learning-Based Corpus Annotation—The PathoJen Experience. *AMIA Annu Symp Proc*, 2012, 301–310. PMID: 23304300, PMCID: PMC3540513.
- Hosseini-Asl, E., Liu, W., & Xiong, C. (2022). A Generative Language Model for Few-shot Aspect-Based Sentiment Analysis. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies: Short Papers (NAACL-HLT 2022) (pp. 770-787). Association for Computational Linguistics.

Hughes, A. (2023, March 16). ChatGPT: Everything you need to know about OpenAI's GPT-4 tool. ScienceFocus. Retrieved from <https://www.sciencefocus.com/future-technology/gpt-3/>

Ipsos. (2018, September 28). *The susceptibility of Singaporeans towards fake news*. Retrieved November 1, 2022, from <https://www.ipsos.com/en-sg/susceptibility-singaporeans-towards-fake-news>

Lee, P., & Raviprakash, S. (2021, October 13). *News in a digital age: Cutting-edge tech meets age-old Virtues & Vices*. Deloitte United Kingdom. Retrieved November 1, 2022, from <https://www2.deloitte.com/uk/en/pages/technology-media-and-telecommunications/articles/digital-consumer-trends-news-sources.html>

Lingren, T., Deleger, L., Molnar, K., Zhai, H., Meinzen-Derr, J., Kaiser, M., Stoutenborough, L., Li, Q., & Solti, I. (2014). Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3), 406-413. <https://doi.org/10.1136/amiajnl-2013-001837>

McKinsey. (2023, January 19). What is generative AI? McKinsey Explainers. Retrieved from <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai>

OpenAI. (2019). Better Language Models and Their Implications. Retrieved from <https://openai.com/research/better-language-models/>

Ouyang, Y. (2020, June 29). *Identifying fake news: The liar dataset and its limitations*. Retrieved November 1, 2022, from <https://towardsdatascience.com/identifying-fake-news-the-liar-dataset-713eca8af6ac>

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. Retrieved from https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

Shinn, N., Labash, B., & Gopinath, A. (2023). Reflexion: An autonomous agent with dynamic memory and self-reflection. arXiv preprint arXiv:2303.11366v1.

Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018). Fever: A large-scale dataset for fact extraction and verification. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. <https://doi.org/10.18653/v1/n18-1074>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. arXiv preprint arXiv:1706.03762. <https://doi.org/10.48550/arXiv.1706.03762>

Vidgen, B., & Derczynski, L. (2020). Directions in abusive language training data: Garbage in, garbage out. arXiv preprint arXiv:2004.01670.

Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (pp. 1112-1122). Association for Computational Linguistics.

Zeng, X., Abumansour, A. S., & Zubiaga, A. (2021). Automated Fact-checking: A survey. *Language and Linguistics Compass*, 15(10). <https://doi.org/10.1111/lnc3.12438>

Appendix A – Final Prompt used for GPT 3.5

You are provided with a claim and evidence sentence, each of which is a string of text. Perform text classification to determine whether the evidence might support, refute, or not have enough information on the claim. Ignore irrelevant elements of the evidence when making judgments about both the temporal and general aspects of the evidence. Only base your decisions on the information explicitly stated in the evidence. Do not assume any information that is not provided. Make sure to differentiate between cases where the evidence is irrelevant or insufficient and cases where the evidence contradicts the claim. Do not infer information that is not explicitly stated in the claim or evidence. Please return the following information in JSON format:

SAME_ENTITY_LABEL: A Boolean value indicating whether the main subject of the claim and evidence refer to the same entity. An entity can be a person, place, or thing. If the claim and evidence refer to the same entity, even if there are coreferences that make it unclear, the SAME_ENTITY_LABEL should be true.

SAME_ENTITY_JUSTIFICATION: A brief justification for your SAME_ENTITY_LABEL.

GENERAL_FACTS_LABEL: Either "SUPPORTS", "REFUTES", or "NOT ENOUGH INFO" indicating whether the non-temporal information in the evidence supports, refutes, or has not enough information to address the claim. If the entire evidence is irrelevant to the claim, does not address all key aspects of the claim, or does not provide any information related to the claim, then the returned label should be 'NOT ENOUGH INFORMATION'. If the evidence directly contradicts any key aspect of the claim, the returned label should be "REFUTES". Do not make assumptions.

GENERAL_FACTS_JUSTIFICATION: An explanation of which part of the evidence supports, refutes, or provides insufficient information for the claim in terms of non-temporal information. If the evidence does not address all aspects of the claim, mention that fact and provide an explanation for why the evidence is insufficient.

TIME_LABEL: Either 'SUPPORTS', 'REFUTES', or 'NOT ENOUGH INFO' indicating whether the date or time information (including exact years) in the

evidence exactly matches the date or time stated in the claim, ignoring general facts and any irrelevant information in the evidence. If the entire evidence is irrelevant to the claim or does not provide any information related to the claim's time, date or duration, then the returned label should be 'NOT ENOUGH INFORMATION'. Do not make assumptions.

TIME_JUSTIFICATION: An explanation for the TIME_LABEL decision based on the date or time information present in the evidence and claim.

TOPIC: The most likely topic that the claim and evidence is referring to. If the claim and evidence are referring to entirely different topics, then return both topics separately.

CLAIM_TEMPORAL: Extract all temporal phrases that are present in the exact phrase of the claim, including phrases that imply a specific time duration, year-based phrases and adverbs or phrases that imply a duration or time period. Temporal phrases can be time, date, or duration. Return a list of strings. Note that each string in the list should be an exact phrase taken directly from the claim.

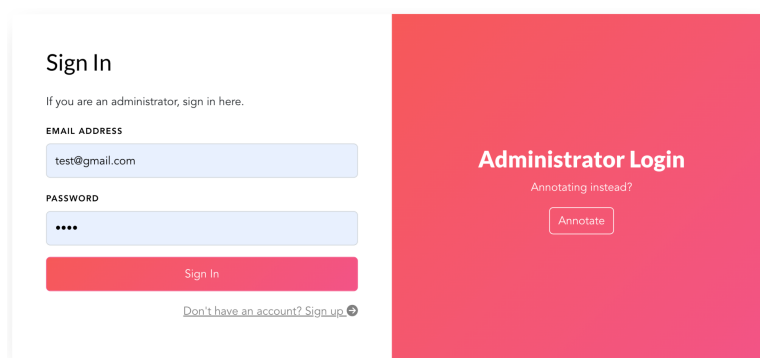
EVIDENCE_TEMPORAL: Extract all temporal phrases that are present in the exact phrase of the evidence including phrases that imply a specific time duration, year-based phrases and adverbs or phrases that imply a duration or time period. Temporal phrases can be time, date, duration, or other phrases that convey a specific point in time or time interval. Return a list of strings. Note that each string in the list should be an exact phrase taken directly from the evidence.

RELEVANT_EVIDENCE: Extract all relevant phrases from the evidence that exactly match any portion of the claim or are semantically similar to any portion of the claim. Relevant parts should be any phrases from the evidence that have the same meaning or convey similar information as the claim or any phrases in the evidence that refer to the same entity as mentioned in the claim, even if they are not exact matches. Return a list of strings. Note that each string in the list must be an exact phrase or sentence taken directly from the evidence.

Appendix B – Annotation Platform

Administrator Dashboard

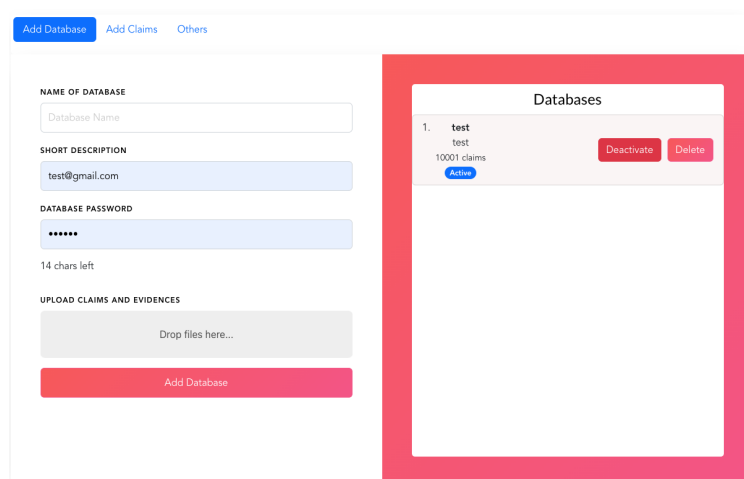
The administrator is responsible for creating the database that contains the claims and evidence that need to be annotated.



The image shows the Administrator Sign-in Page. It is split into two main sections. The left section is white and contains a 'Sign In' form. The right section is a solid red color with white text. The form on the left has a title 'Sign In', a subtitle 'If you are an administrator, sign in here.', and two input fields: 'EMAIL ADDRESS' with the value 'test@gmail.com' and 'PASSWORD' with masked characters '****'. Below the password field is a red 'Sign In' button. At the bottom of the form is a link 'Don't have an account? Sign up' with an external link icon. The red section on the right has the title 'Administrator Login', the subtitle 'Annotating instead?', and a white 'Annotate' button.

Administrator Sign-in Page

The administrator will go through a sign-up and login process in order to create an user account.



The image shows the Dashboard to create a database. It has a top navigation bar with three tabs: 'Add Database' (active), 'Add Claims', and 'Others'. The main content area is split into two columns. The left column is white and contains a form to create a new database. The right column is a solid red color and contains a 'Databases' table. The form on the left has four sections: 'NAME OF DATABASE' with a text input field, 'SHORT DESCRIPTION' with a text input field containing 'test@gmail.com', 'DATABASE PASSWORD' with a masked password field and a '14 chars left' indicator, and 'UPLOAD CLAIMS AND EVIDENCES' with a 'Drop files here...' area and an 'Add Database' button. The 'Databases' table on the right has a header 'Databases' and a single row with the following data: '1.', 'test', 'test', '10001 claims', 'Active' (with a blue 'Active' button), 'Deactivate' (red button), and 'Delete' (red button).

Dashboard to create a database

Once the administrator signs in, he will be able to conduct CRUD (create, read, update, delete) operations on the databases, claims and evidence. Claims and evidence can be easily uploaded via a json file, as that is typically the output of the previous data retrieval process. The database can also be activated for labelling.

Add Database
Add Claims
Others

SELECT A DATABASE
53 - test +
☐ Check to upload files instead

CLAIM

ORIGINAL ID

ADD EVIDENCE
No evidence added yet.

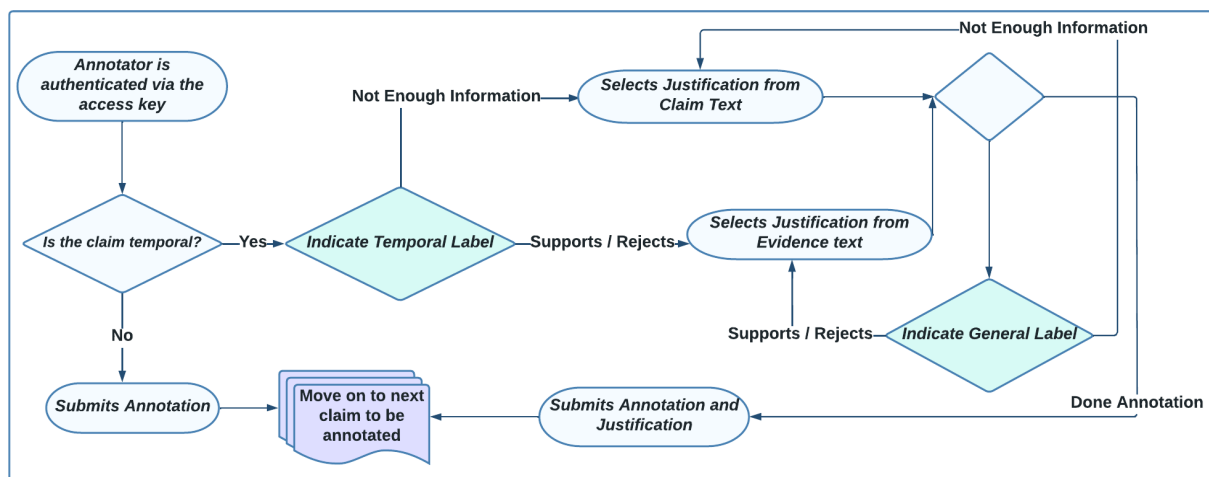
Claims

1. System of a Down briefly disbanded in limbo.	<input type="button" value="Delete"/>
2. Beautiful reached number two on the Billboard Hot 100 in 2003.	<input type="button" value="Delete"/>
3. Neal Schon was named in 1954.	<input type="button" value="Delete"/>
4. Cyndi Lauper won the Best New Artist award at the 27th Grammy Awards in 1985.	<input type="button" value="Delete"/>
5. Robert J. O'Neill was born April 10, 1976.	<input type="button" value="Delete"/>
6. Peggy Sue Got Married is a Egyptian film released in 1986.	<input type="button" value="Delete"/>
7. Andy Roddick lost 5 Master Series between 2002 and 2010.	<input type="button" value="Delete"/>
8. As the Vietnam War raged in 1969, Yoko Ono and her husband John Lennon did not have two week-long Bed-Ins for Peace.	<input type="button" value="Delete"/>

Dashboard to manually add claims and evidences

A manual form for adding claims and evidence is also provisioned.

Annotator



Activity flow for annotation

Stepwise Flow (With UI)

Annotate

If you are an annotator, access your annotation tasks here.

DATABASE ID

This should be an unique number.

ANNOTATION PASSWORD

....

ANNOTATOR EMAIL ADDRESS

Please fill in your unique email address to identify you.

Start Annotation

Annotator Access

Administrator?

Go to Administrator Sign In

Annotator Access Page

The annotator first accesses the homepage and inputs the database id and annotation password (access key) that should be provided by the database administrator. The annotator must also input his/her email address for identification purposes.

Annotating Database: 53

10001 claim

Back to Home

Annotator: test@gmail.com

Topic: System of a Down

Temporal parts of the claim and evidence are highlighted in yellow.

Claim

System of a Down briefly disbanded in limbo.

Evidence (1 out of 2)

Svstem of a Down is the debut studio album by Armenian American metal band System of a Down, released on June 30, 1998, by American Recordings and Columbia Records.

Temporal Label

Supports	Refutes	Not Enough Information
Please select a temporal label before moving on to temporal justification.		

Next to temporal justification


[Report a problem](#)

Annotator Annotation Page

Once the annotator is successfully authenticated, he will be given a claim and evidence pair that he has not previously annotated before. He is then in the annotation process, where he has to first select if the temporal elements of the evidence support the temporal elements of the claim. Besides this, the annotator has to provide justification by using his mouse to select the portions of the claim or evidence that lead to the final labels. The annotation page is designed as clean, and easy to use, as possible.

Evidence (1 out of 2)
System of a Down is the debut studio album by Armenian American metal band System of a Down, released on **June 30, 1998**, by American Recordings and Columbia Records.

Justification for Temporal Label
You have selected "Supports" for the temporal classification. Please use your mouse to select which temporal part of the evidence texts support the temporal parts of the claim.

June 30, 1998 

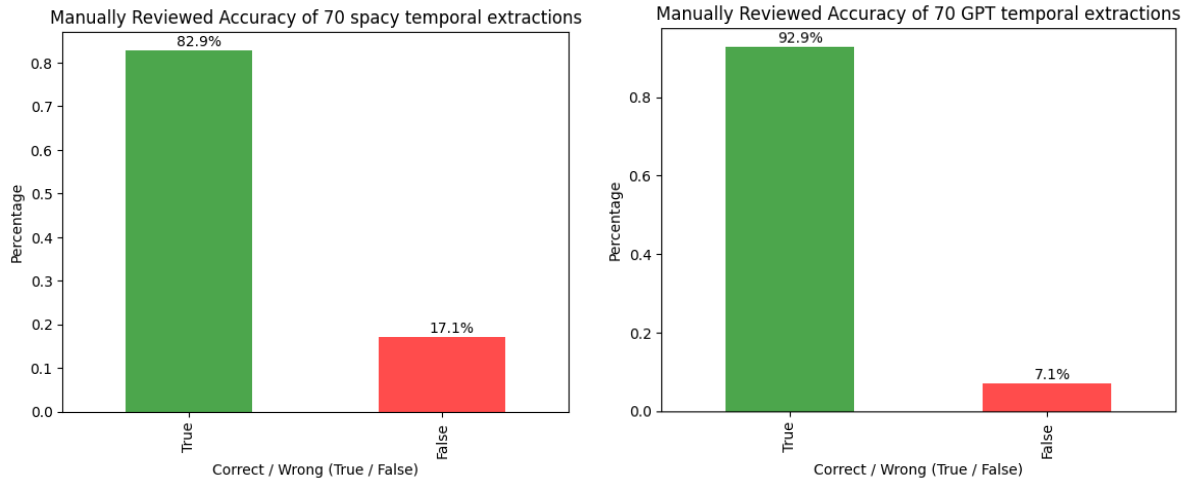
Back to temporal label **Next to general label**

Label Justification

The selected elements are automatically added to the justification list, and the annotator can add multiple pieces of justification. If the labels selected are “Not enough information”, then the annotator has to select the part of the claim that lacks justification from the current evidence source. Otherwise, “Supports” or “Refutes” labels both require the annotator to select justification from the evidence source. The annotation system automatically displays the next claim once the annotator has completed the current annotation.

GPT’s role in highlighting temporal and relevant elements

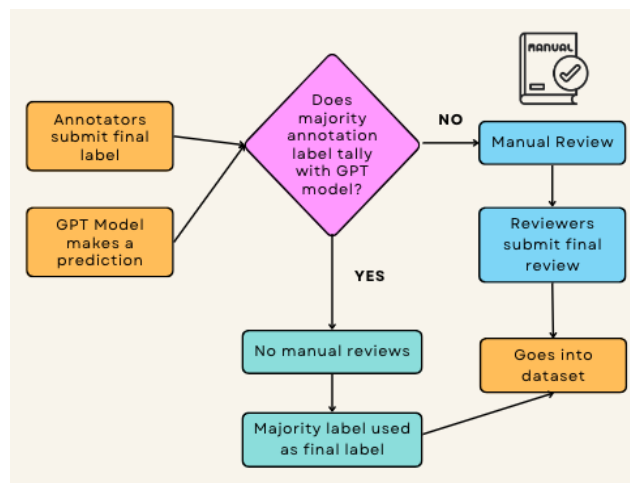
In order to reduce the amount of time that annotators spend on identifying the temporal label, we harness the predictive abilities of GPT to extract the temporal elements of both claim and evidence sentences. These elements are then highlighted to the annotator so that it becomes much easier for annotators to make comparisons.



Evaluation of GPT's ability to extract temporal labels

In a brief evaluation of GPT's ability to extract temporal labels, we compared to an existing popular package Spacy on 70 manually labelled FEVER claim entries. GPT outperforms SPACY in correctly and comprehensively identifying temporal elements of the claim 92.9% of the time, compared to Spacy's 82.9%. This is likely because Spacy fails to identify phrases that relate to time, such as "briefly", "usually" or "during the holocaust". GPT is more contextually aware, so these temporal references are also successfully captured. When the annotators move on to annotate the general label, GPT then holds the relevant portions of the evidence, so that annotators can refer to the portions of the evidence that would most likely influence the general label.

Manual Review



How annotation entries are chosen for manual review

For the manual review phase, the system tallies all annotations done for the same claim and evidence pair, and looks at the majority annotated label. If the majority label tallies with the

prediction made by GPT, then the entry is automatically given its final label, and no manual review is required. This prioritises manual reviews for entries which have a higher likelihood of being wrong, and adopts a structured approach to identifying such entries, rather than doing a random sample. This approach also ensures that every entry is “inspected”. Though both the annotators and the GPT model do not have a perfect accuracy as standalone systems, they can complement each other, and such a combination would result in a lower probability of annotation errors compared to either the annotators or GPT alone. This is because the errors made by the annotators and GPT are highly likely to be independent of each other. This is also why we do not display the GPT prediction results to the annotator during the annotation process to guide the annotator, because there is a chance that the annotator could be unduly reliant on the model’s predictions, and fail to come to his or her own conclusions on the correct labels.

[Claim](#)

Beautiful reached number two on the Billboard Hot 100 in 2003.

The song peaked at number two on the Billboard Hot 100 in the United States , where it was certified Gold for 500,000 units shipped .

Annotation Done:

0

Supports

1

Refutes

0

Not enough info

1. test@gmail.com

0 % ignored

03/04/2023, 17:01:45

General Label: REFUTES

Temporal Label: REFUTES

Final Label: **REFUTES**

Ignore this annotation

Ignore all

GPT's Predictions

General Label: **SUPPORTS**

The evidence confirms that 'Beautiful' reached number two on the Billboard Hot 100 and was certified Gold for 500,000 units shipped, which supports the claim.

Temporal Label: **SUPPORTS**

The evidence states that 'Beautiful' peaked at number two on the Billboard Hot 100 in the United States, which matches the claim that it reached number two on the Billboard Hot 100 in 2003.

Final Model Label: **SUPPORTS**

Submit Final Label

Select a final label

Next Review

The Manual Review Page

In the manual review section, investigators can see both the GPT’s predictions, as well as the annotator’s annotations. GPT justifies its own label decisions, so this provides additional

explainability to assist the investigator in the manual review process. It is also possible for the investigators to ignore specific annotation entries. Investigators select a final label at this stage, by adjudicating both the model's predictions, as well as the majority annotation label.