

國立臺灣大學電機資訊學院資訊工程研究所
碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

推文中的宣稱檢測

Claim Detection in Tweets

鄒詠霖

Yung-Lin Tsou

指導教授：許永真 博士

Advisor: Jane Yung-jen Hsu, Ph.D.

中華民國 112 年 7 月

July, 2023

Acknowledgments

首先想要感謝我的指導教授許永真博士給予的尊尊教誨，讓我從原本對研究要如何進行都不太清楚也沒有非常明確的目標到現在可以完成自己的研究。每次與教授開會都受益良多，不管是技術方面或者是做人處世的態度上面，都從教授這邊學習到了非常多。接著，我想要感謝口試委員(蔡宗翰教授、郭彥伶教授、古倫維博士、李育杰教授)在口試時給予我的專業建議以及點出我研究不足的地方，得以使我的研究更完善。再來，我想要感謝實驗室的同學們，與大家相處得非常愉快，也從同學們身上學習到很多不同的知識。最後，我想感謝我的家人對於我的支持。

Abstract

With the advancement of Internet technology, the issue of misinformation is increasingly serious. Among the many methods of dealing with misinformation, fact-checking is considered the most trustworthy means. However, the traditional fact-checking process, which verifies the authenticity of information manually, may be too time-consuming to keep up with the speed of information generation. Thus, the automation of fact-checking has become a highly valuable goal. Automated fact-checking is generally divided into four stages, including claim detection, verified claim retrieval, evidence retrieval, and claim verification.

This study focuses on solving claim detection in tweets. The goal of claim detection is to determine which part of the content is a claim, and a claim can be defined as *"an assertion of something as a fact"*. Verifying the truth of these claims is the process of fact-checking. In this research, we first propose two issues that have not been adequately addressed in claim detection within tweets : tweet's noisy format and task definition, and propose corresponding methods to solve these problems. In addition, we also validate the effectiveness of our proposed methods through experiments and summarize the insights and findings from the experiments.

Keywords: Automatic Fact-checking, Claim Detection, Natural Language Processing

摘要

隨著網際網路技術的進步，假信息的議題日益嚴重。在眾多處理假信息的方法中，事實查核被視為最值得信任的手段。然而，傳統的事實查核通過人力驗證信息真偽的過程可能耗時過長，無法趕上信息產生的速度，因此事實查核的自動化成為了了一個極具價值的目標。自動化事實查核通常分為四個階段，包括宣稱檢測、已驗證的宣稱檢索、證據檢索、與宣稱驗證。

本研究著重於解決推文中的宣稱檢測問題。宣稱檢測的目標是確定哪部分內容是宣稱，而一個宣稱可被定義為「宣稱某事是事實」，驗證這些宣稱的真偽即為事實查核的過程。在這項研究中，我們首先提出了過去在推文中進行宣稱檢測時兩個未被充分處理的問題：推文的混亂格式與任務定義，並提出了相應的方法來解決這些問題。此外，我們也透過實驗去驗證我提出的方法的有效性，並總結了實驗中的啟示與發現。

關鍵字：自動化事實查核, 宣稱檢測, 自然語言處理

Contents

Acknowledgments	i
Abstract	ii
摘要	iii
List of Figures	vii
List of Tables	x
Chapter 1 Introduction	1
Chapter 2 Related Work	7
2.1 Style-Based Misinformation Detection	7
2.2 Automatic Fact Checking	8
2.3 Claim Detection	8
2.4 Language Model(LM)	9
2.5 Semantic Role Labeling(SRL)	10
2.6 Bidirectional Encoder Representations from Transformers(BERT) . .	11
2.7 Generative Pre-trained Transformer(GPT)	12

2.8	Text Style Transfer	13
Chapter 3	Task Definition, Dataset & Current Approach	14
3.1	Task Definition	14
3.2	Article-Level Dataset	15
3.3	Sentence-Level Dataset	16
3.4	Current Approach	18
Chapter 4	Research Questions	22
4.1	Tweet’s Format	22
4.2	Improper Task Definition	24
Chapter 5	Methodology	26
5.1	Reducing Noise in the Formatting of Tweets	27
5.2	Reducing the Inconsistency between the Level of Input and Claim . .	29
5.3	Combine All	33
Chapter 6	Experiments	38
6.1	Experimental Setting & Evaluation Metric	38
6.2	Elimination of Special Symbols and URLs	39
6.3	Different Prompt when Rewriting Tweets	43
6.4	Split Tweets into Sentence-Level	48
6.5	Train with Sentence-Level Dataset	52
6.6	Combined All	55
Chapter 7	Conclusion	59

List of Figures

1.1	Tweet generation per day from 2006 to 2013, as calculated by Internet Live Stats. Because we couldn't find any other statistics containing more recent data that are equally trustworthy, we used this older statistics.	2
1.2	Automatic Fact-Checking Pipeline	4
1.3	Social Media Daily Usage in 2021. The number means the percentage of user.	6
2.1	SRL example. The words colored in blue and red represent two semantics extracted by SRL. The circled words are the predicates, and the words pointed to by the predicate are the arguments of that semantic.	11
3.1	Structure that most of the current approach adopt	19
3.2	Check_square Structure[11]	20
3.3	LESA Model Structure[12]	21

4.1	An example of the noisy symbols in tweets. The content highlighted in red represents the symbols and URLs that create the "noise" in a tweet.	23
4.2	An example of the informal writing style in tweets. The content highlighted in red represents the informal writing style that create the "noise" in a tweet.	23
4.3	An example of inconsistency between tweet-level and claim-level. The content highlighted in red includes two claims present within this tweet, in our opinion.	25
5.1	An example of eliminating the special symbols and URL in a tweet .	27
5.2	An example of a tweet rewrite by GPT using prompt "rewrite:". . . .	28
5.3	Pipeline for dealing with tweet's noisy format. The Model mentioned here can be refer to the Model in Figure 3.1.	28
5.4	Result of rule-based splitting the tweet	30
5.5	Result of using SRL extract sentences in the tweet	30
5.6	An example of combine the predictions from sentence-level back to article-level.	31
5.7	Pipeline for dealing with improper task definition. The Model mentioned here can be refer to the Model in Figure 3.1.	32
5.8	An example showing the original splitting result by AllenNLP SRL tool.	34
5.9	Pipeline for dealing two problem simultaneously. The Model mentioned here can be refer to the Model in Figure 3.1.	36
5.10	An example of a tweet process through rewrite by GPT then split into sentence-level.	37

6.1	The difference between Delete '#' '@' , and Delete tail hashtag and tagging . The left example is Delete '#' '@' , which literally deletes the symbols '#' and '@', but retains the text attached to them. The right example is Delete tail hashtag and tagging , which aims to delete those hashtags and taggings at the end of a tweet by deleting both the symbol and the text attached.	41
6.2	Nonsense user name example.	42
6.3	A success case which turn the wrong prediction to right after rewriting.	44
6.4	A tweet rewrite by GPT using "rephrase" as prompt. The content highlighted by red were not present in the given tweet.	45
6.5	Same tweet rewrite by GPT with different prompt	46
6.6	A more detailed prompt we elaborate to rewrite the tweet.	47
6.7	A prompt we elaborate to rewrite the tweet which using clearly guide-lines.	48
6.8	An example demonstrating how we prompt GPT to directly classify whether the tweet contain any claims.	56

List of Tables

3.1	Article-Level Datasets	16
3.2	Sentence-Level Datasets	17
6.1	Hyperparameter and experimental setting.	39
6.2	Experiment result of deleting special symbol and URL.	40
6.3	Experiment result of different prompt. The prompt is using as the part highlighted in blue in Figure 6.4.	43
6.4	Experiment result of different split method	49
6.5	Example of the prediction after splitting. The text highlighted in red is the claim present in this tweet in our opinion.	50
6.6	An example of the prediction after splitting. The text highlighted in red and blue is two claims present in this tweet in our opinion.	51
6.7	Experiment result of different training set	53
6.8	Experiment result of combined all proposed method. The Model men- tioned here can be refer to the Model in figure 3.1	57

Chapter 1

Introduction

Thanks to the flourishing of the Internet and social media, obtaining information nowadays has become very easy. Although this brings a lot of convenience, it also gives rise to a significant problem associated with this convenience: how can we assess the accuracy of the information we obtain? This becomes a major issue when the information we receive could be **misinformation**.

Misinformation can be defined as *"false information that is spread, regardless of whether there is intent to mislead"*¹. Misinformation flooding on the Internet and social media has become a significant issue as it misleads those who believe in it and diminishes people's trust in online platforms. To address this, fact-checking comes into play. Fact-checking can be referred to as the action of verifying the truth of a claim. By fact-checking all the claims in an article, its credibility can be evaluated and its categorization as misinformation or not can be more accurately determined. To date, there are many organizations such as the Taiwan FactCheck

¹<https://www.dictionary.com/browse/misinformation>

Center², FactCheck.org³, Snopes⁴, PolitiFact⁵, and FullFact⁶ worldwide that are dedicated to the task of fact-checking. The Duke Reporters LAB⁷ maintains a website that lists all of the active fact-checking organizations around the world, currently contain approximately 400 organizations.

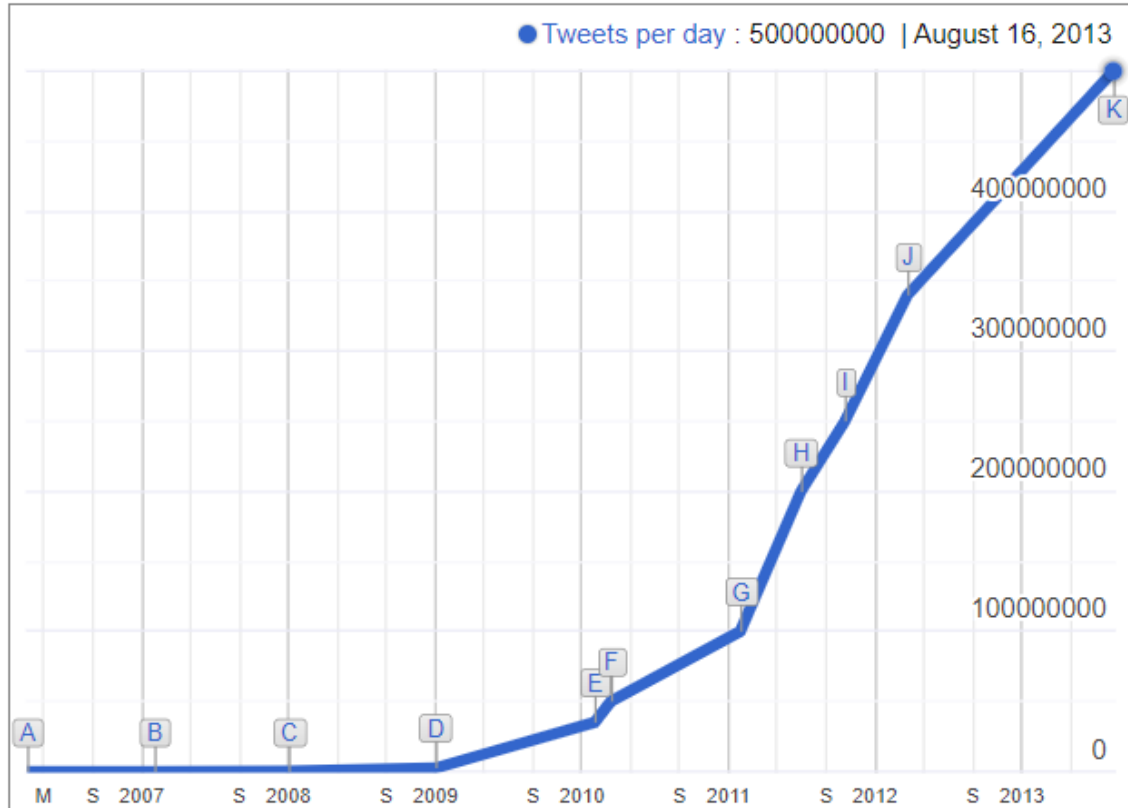


Figure 1.1: Tweet generation per day from 2006 to 2013, as calculated by Internet Live Stats. Because we couldn't find any other statistics containing more recent data that are equally trustworthy, we used this older statistics.

²<https://tfc-taiwan.org.tw/>

³<https://www.factcheck.org/>

⁴<https://www.snopes.com/>

⁵<https://www.politifact.com/>

⁶<https://fullfact.org/>

⁷<https://reporterslab.org/fact-checking/>

Despite the existence of fact-check organizations, the capability of humans to fact-check can't keep pace with the proliferation of misinformation. Figure 1.1 shows the number of tweets generated per day from 2006 to 2013, as calculated by Internet Live Stats⁸. In 2013, about 500 million tweets were generated per day. To understand the rate of fact-checking, consider the daily fact-check report number published on PolitiFact, which typically does not exceed 10. Although we can't determine the actual number of misinformation instances, these figures suggest that the throughput of manual fact-checking is grossly insufficient. Therefore, automating the fact-checking process is necessary to handle the vast amount of information.

Manual fact-checking practices may vary among different fact checkers due to personal habits, but for machines, a clear process is required. A. Barron-Ced, et al. proposed a pipeline for automatic fact-checking in their work introducing the competition they organized[3]. This pipeline divides fact-checking into four sub-tasks that can be addressed independently. Figure 1.2 shows the pipeline we derived from theirs, which consists of claim detection, verified claim retrieval, evidence retrieval, and claim verification. The claim detection stage aims to identify any claims within an article. Next, the verified claim retrieval stage determines whether the detected claims have already been checked; if they have, the verification of that claim can be directly obtained from historical data and the pipeline stops; otherwise, the next stage continues. The third stage involves retrieving the evidence needed to assess whether the claim is true or false. Finally, with the gathered evidence, verdicts for those claims can be formulated. Based on these verdicts, the original article can be categorized as misinformation or not.

⁸<https://www.internetlivestats.com/twitter-statistics/>

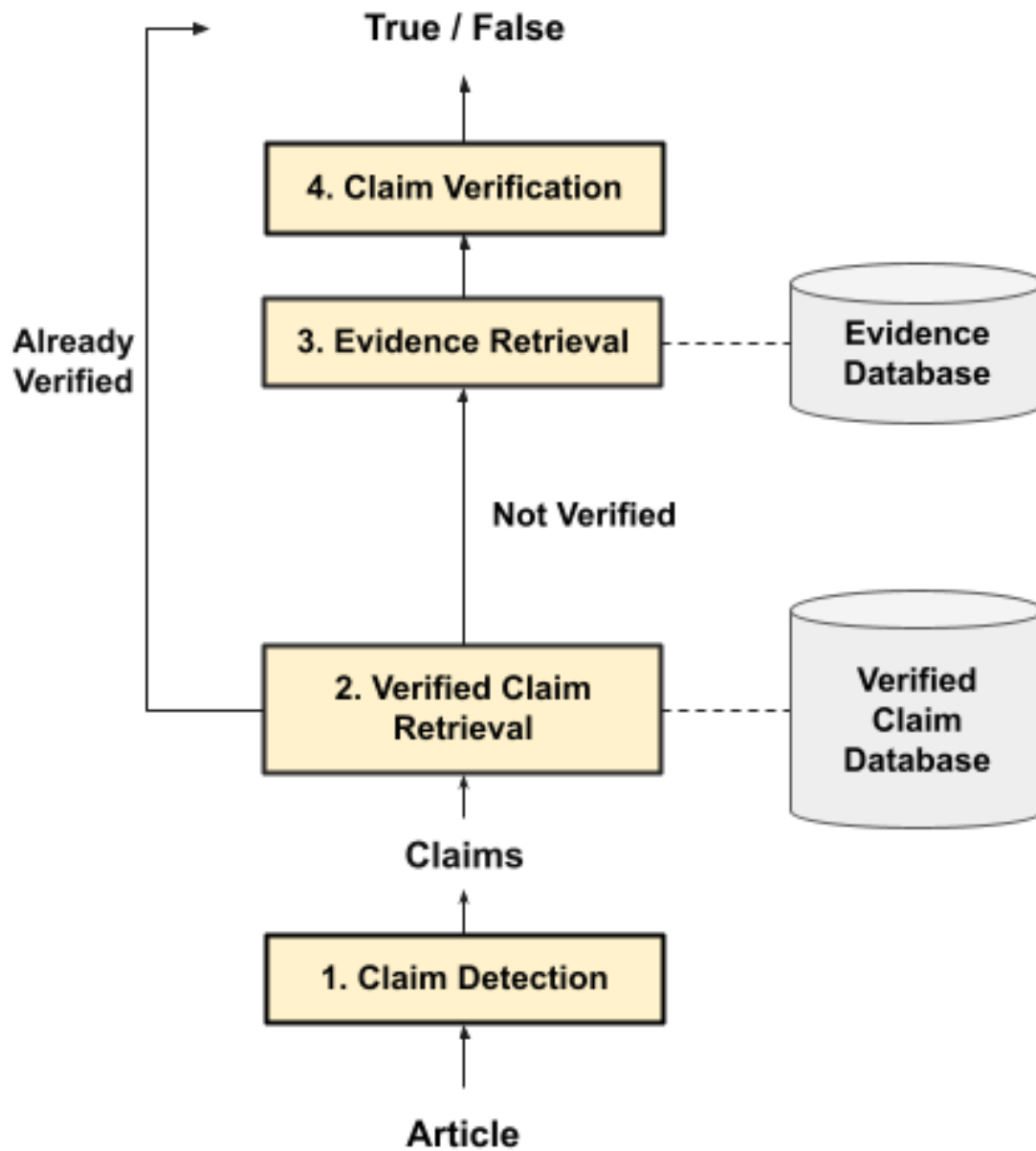


Figure 1.2: Automatic Fact-Checking Pipeline

Although fully automated fact-checking seems ideal, the trustworthiness of machine-derived verdicts is still a matter of debate. Research conducted by FullFact[23]

reveals that a majority of expert fact-checkers do not yet trust verdicts made by machines. According to their study on the needs of fact-checkers, the process of automatic fact-checking should conduct by partial automating the pipeline to assist fact-checkers, rather than fully automating the entire process and allowing machines to render verdicts about the veracity of information.

In this work, we concentrate on automating the claim detection stage because we believe this degree of automation is acceptable by fact-checkers and could really integrating into their fact-checking process. A **claim** can be defined as *"an assertion of something as a fact"*⁹. With a proper definition of claim and a suitably annotated dataset, we can employ machine learning techniques to automate claim detection.

On the other hand, from a data source perspective, we are primarily concerned with addressing misinformation on social media. Unlike articles on news media, users on social media can post articles with almost no restrictions and without a rigorous review process. This freedom makes social media a fertile ground for the propagation of misinformation. Another reason for focusing on social media is its pervasive usage. Figure 1.3 provides statistics from Pew Research¹⁰ depicting the daily usage of 1500 adults in the US. As seen, more than half of these users access social media at least once a day. With higher engagement comes an increased risk of misinformation absorption. Therefore, tackling misinformation on social media is both urgent and valuable. Lastly, our research is primarily focused on tweets, as most existing datasets aiming at claim detection on social media gather their data from Twitter.

While there exist several methods to achieve this task, we observe that some problems have not been adequately addressed, which will be detailed in Chapter 4.

⁹<https://www.dictionary.com/browse/claim>

¹⁰A survey about social media use

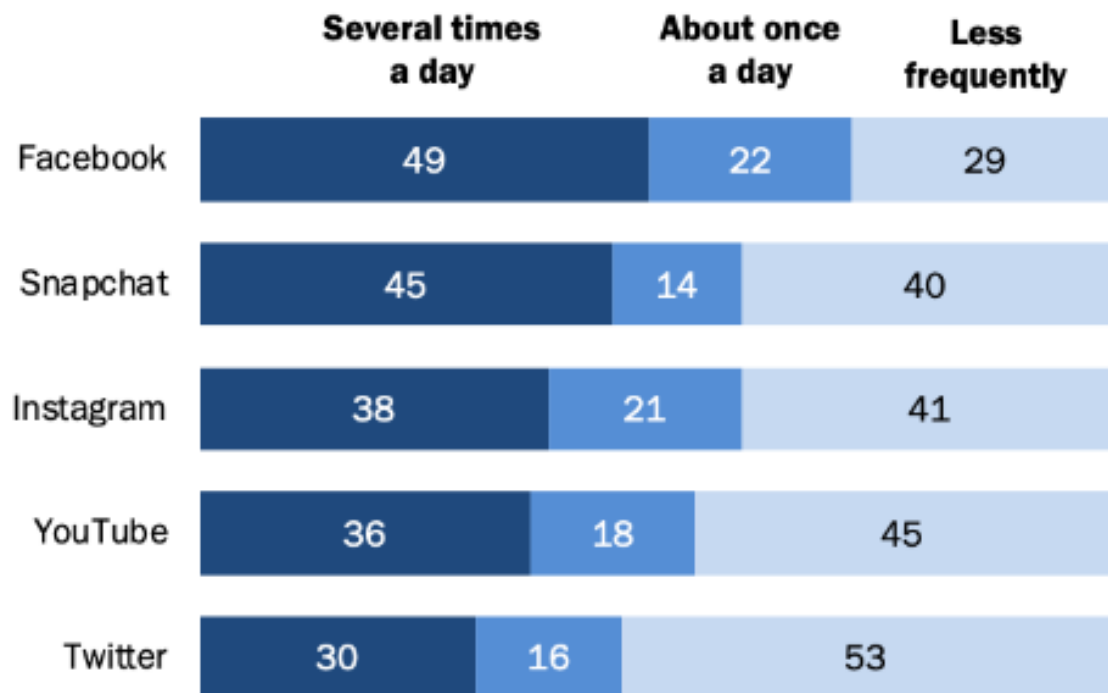


Figure 1.3: Social Media Daily Usage in 2021. The number means the percentage of user.

Therefore, the objective of this research is to propose methodologies aimed at resolving these problems, which are anticipated to further improve performance on this task.

Chapter 2

Related Work

In this section, we'll first introduce the work related to misinformation. Subsequently, we'll provide an overview of the models and techniques utilized in this research.

2.1 Style-Based Misinformation Detection

In response to the growing volume of misinformation, increasing amounts of research are being dedicated to its automated management. The simplest approach involves classifying an article as misinformation based solely on its writing style, a method we refer to as **style-based misinformation detection**. A standard procedure for constructing a style-based misinformation detection system entails training a machine learning model on a dataset composed of articles labeled either as misinformation or not misinformation[25]. This approach hinges on the robust assumption that the writing style of articles containing misinformation differs from those that do not. This assumption is plausible as some misinformation indeed employs specific writing styles, such as exaggeration or sensationalism, to seize public

attention. However, it is neither reasonable nor persuasive to presume all misinformation follows this pattern. Consequently, automatic fact-checking appears to be a more credible and accepted method to combat misinformation effectively.

2.2 Automatic Fact Checking

The fact-checking approach tackles misinformation by verifying the veracity of the claim being made. After reaching a verdict, it provides evidence to either support or refute these claims, making it a more convincing method compared to style-based misinformation detection.

A significant amount of research in automatic fact-checking focuses on the first stage(claim detection) or the fourth stage(claim verification)[33, 34], as these tasks are relatively new. Stages two(verified claim retrieval) and three(evidence retrieval) resemble information retrieval, a well-studied topic, and are therefore perceived as less novel for research. These stages can refer to Figer 1.2. Some work also attempts to cover the entire fact-checking pipeline. For instance, in [13], the authors propose a pipelined system called ClaimBuster, which is capable of performing end-to-end fact-checking without the need for human intervention.

2.3 Claim Detection

Prior to fact-checking, claim detection had already emerged as a research topic in argument mining. Back in 1958, Toulmin proposed an argument model[35] that deconstructed arguments into six parts. This model has been widely used in argument mining. The six parts include the claim, grounds, warrant, qualifier, rebuttal, and backing. Among them, the claim is the most important component as it represents

the assertion the author aims to prove. With this argument model in place, argument mining became a task that sought to automatically extract these parts of an argument[17]. Thus, claim detection naturally appeared as a subtask of argument mining[1, 18, 19]. Later in fact-checking, Toulmin’s definition of a claim continued to be referenced by many[6, 12].

Claim detection plays a pivotal role as the initial step in fact-checking. If the outcome of claim detection is inaccurate, the subsequent stages of fact-checking will be a futile labor since the verdict on a non-claim is meaningless and, therefore, cannot contribute to judging the authenticity of the entire article. From this perspective, the accuracy of claim detection is critical to the success of automated fact-checking.

2.4 Language Model(LM)

Language models refer to those models that aim to understand the natural language used by humans. The research on language models has a long history, dating back to the emergence of the word n-gram in a paper written by C.Shannon in 1948[31]. However, the capabilities of language models were not significantly impressive until the rise of neural networks. With the advent of these networks, many attempted to build language models using different network types[4, 8].

The state-of-the-art language models are mainly established by the attention mechanism proposed by Ashish Vaswani et al. in 2017[36] and self-supervised learning that leverages the power of massive unlabeled texts online. Currently, there are two mainstream self-supervised training objectives for constructing attention-based neural language models. The first objective involves sequentially predicting the next word given a string, with the most famous model of this type being GPT[26, 27, 5]. The second approach involves predicting the word that is masked in the given string,

for which BERT[7] is the most representative one.

Recently, language models have become larger, resulting in the emergence of a new term, Large Language Model (LLM). LLMs refer to those language models containing over a billion parameters. Among these, GPT has been the most recognized since its second version.

2.5 Semantic Role Labeling(SRL)

SRL aims to extract all the semantics within a given paragraph of text. Semantic is an essential unit that expresses meaning when communicating via natural language by providing the information “Who did What to Whom(When and Where)”[16]. Each semantic consists of one predicate that points out “What” in this semantic and the rest of the component is called argument which points out “Who”, “Whom”, “When” and “Where”. Usually, predicate is a verb and argument is the noun around that verb that is related to it. The history of Semantic Role Labeling(SRL) can be traced back to 1968, when Charles J. Fillmore proposed to build the project FrameNet[2] which constructed the first dataset for SRL. In FrameNet, they refer each semantic as a “frame” and this synonym has also been widely used when referring to semantics that are extracted by a SRL model.

To achieve SRL, the most commonly used approach is to first extract all the predicates from the given input. A predicate is an essential component that forms a semantic. These predicates are usually verbs, making their identification relatively straightforward. Then, for each predicate, the next step involves tagging each of the remaining words. This tag specifies whether the word is related to the predicate and, if so, how it is related. Finally, one predicate and all its related words together form a semantic. Figure 2.1 shows an example of a sentence and its SRL result. The

state-of-the-art method for SRL currently involves using a language model(LM) as an encoder to transform each word into an embedding. This word embedding is then used to predict its semantic role label[9].

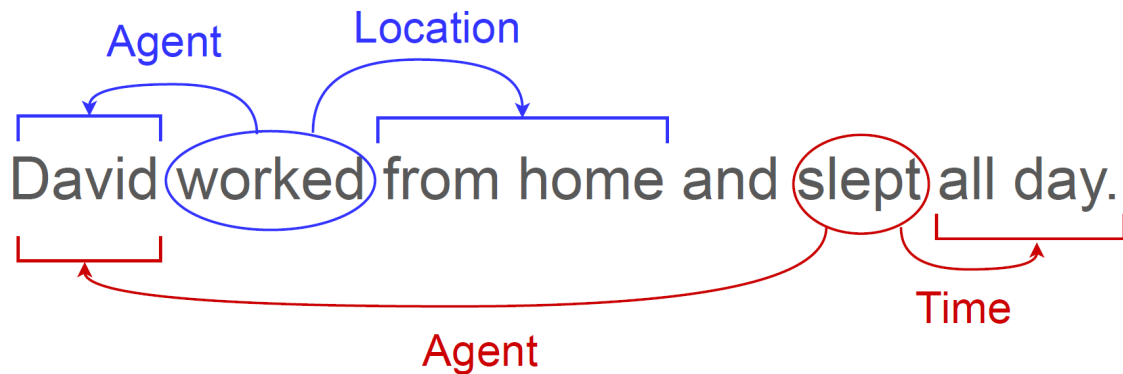


Figure 2.1: SRL example. The words colored in blue and red represent two semantics extracted by SRL. The circled words are the predicates, and the words pointed to by the predicate are the arguments of that semantic.

2.6 Bidirectional Encoder Representations from Transformers(BERT)

Going back to 2018, when Google first proposed the Bidirectional Encoder Representations from Transformers (BERT) model[7], it quickly seized the attention of the NLP community and has since become widely adopted. In BERT, two training objectives - Masked Language Model (MLM) and Next Sentence Prediction (NSP) - are used to pre-train the model with massive amounts of unlabeled online content. The bidirectional model architecture, coupled with the MLM objective, enables BERT to excel at understanding text by generating meaningful embedding vectors. This

feature makes BERT a strong model for solving tasks such as classification.

Many variations of BERT have been proposed to optimize its performance in various contexts. For example, RoBERTa[20] alters the pre-training procedure to enhance stability. These continuous advancements have led to the widespread adoption of BERT-based models to date.

2.7 Generative Pre-trained Transformer(GPT)

Since 2018, when OpenAI introduced the concept of the Generative Pre-trained Transformer(GPT)[26], it has continuously updated its model. To date, four versions have been released. Beginning with version 2[27], the model size of GPT expanded to the order of billions, featuring 1.2 billion parameters, which qualifies it as a Large Language Model(LLM). In GPT-3[5], the model size was further increased to an astounding 175 billion parameters. This growth in model size has been proven effective, as evidenced by the steadily improving performance of the GPT series.

The pre-training method adopted by GPT(i.e., sequentially predicting the next word) naturally makes it an effective language generator. Its prominent language capabilities, showcased in ChatGPT(an application developed by OpenAI for conversing with GPT), also contributed to its widespread recognition. These capabilities enable GPT to be applied to any downstream NLP task by converting the task into a question-answering format. The question asked to facilitate GPT’s understanding of the downstream task is commonly referred to as a “prompt,” and the tuning of prompts has recently become a popular research topic.

2.8 Text Style Transfer

Text Style Transfer (TST) is a task that aims to transfer the original writing style of content to a specific writing style, such as converting informal language to formal[28], impolite to polite[21], or offensive to non-offensive[24], etc. This task can be treated as a natural language generation task where, given a paragraph of natural language, the output is also a paragraph of natural language. Most state-of-the-art methods that achieve text style transfer use paired data, which contains the same content written in two different writing styles. This kind of data is used to train a language model that guides the transition from one writing style to another. TST can be applied to many kinds of specific downstream tasks[15], such as making the content more attractive and engaging, or anonymizing user identity by obfuscating their writing style, etc.

Chapter 3

Task Definition, Dataset & Current Approach

In this chapter, we will first define the task of claim detection in tweets that we aim to solve. Then, we will introduce some existing datasets, compare them, and identify the ones used in this research. Lastly, we will discuss some of the current approaches employed to accomplish this task

3.1 Task Definition

The definition of claim detection we are trying to solve is as follows:

Definition 3.1.1 (Claim Detection) *Given a tweet, determine whether it contains any claims.*

Therefore, the input is a tweet, and the output should be a binary value. A value of 0 represents that there are no claims in the tweet, while a value of 1 represents that there is at least one claim made in the tweet.

3.2 Article-Level Dataset

Table 3.1 shows three existing claim detection datasets that fit the definition described above. **LESA** is the model abbreviation proposed by S. Gupta et al.[12]. Although their work primarily focuses on proposing a model architecture that better deals with claim detection regardless of the data source, they also built a dataset sourced from Twitter. In building the dataset, their definition of a claim to guide annotators is *"state or assert that something is the case, with or without providing evidence or proof."*

CheckThatLab! is a competition held by CLEF(Conference and Labs of the Evaluation Forum), which focuses on tasks related to misinformation. Since 2018, they have conducted this competition annually, and we chose to use the dataset they released in 2022. In 2022, they proposed several subtasks related to misinformation, and among them, subtask 1b is the dataset that best fits the definition of claim detection we provided. The claim they aim to detect adds an additional condition that it should be verifiable. According to their definition, a verifiable factual claim is *"a sentence claiming something to be true, and this can be verified using factual verifiable information such as statistics, specific examples, or personal testimony."*

Lastly, **BioClaim** is a dataset focused on tweets related to four diseases, which was proposed in the work by A. Wuhrl, et al.[37]. The claim definition they use is *"the argumentative component in which the speaker or writer expresses the central, controversial conclusion of their argument."*

Since the amount of data in **BioClaim** is significantly limited, we only utilized the first two datasets, **LESA** and **CheckThatLab! 2022 task 1b**, to verify the method we proposed.

	# Data	Avg length	Data Source	Topic	Output Format	Percentage of claim
LESA[12]	9981 tweets	26.36	Twitter	covid-19	Binary	87.43%
CheckThatLab! 2022 task 1b[22]	4793 tweets	37.56	Twitter	covid-19	Binary	63.43%
BioClaim[37]	1200 tweets	31.23	Twitter	Disease	Binary	44.75%

Table 3.1: Article-Level Datasets

3.3 Sentence-Level Dataset

Other than the dataset at the article-level, there are additional sentence-level datasets available, as shown in Table 3.2. In the ClaimSpotter module of **ClaimBuster**[13], they train their claim detection model using a dataset that gathers US presidential debates from 1960 to 2012. In their dataset, they treat every sentence stated by each candidate as individual data points. **NewsClaims**[29] is a dataset sourced from news articles. In addition to claim detection, they also define other tasks such as detecting the claimer and identifying the object being claimed, etc. **VG**, **WD**, **PE**, **OC**, **WTP**, and **MT** are six datasets released in [6]. Originally, these six datasets were not dedicated to claim detection. Instead, most of them were proposed to handle argument mining. However, as claim detection is an important component of argument mining, the author of [6] converted them into sentence-level claim detection datasets.

Although the sentence-level and article-level datasets both aim to achieve claim detection, their meanings differ due to the input level. In the article-level dataset,

	# Data	Avg length	Data Source	Output Format	Percentage of claim
ClaimBuster[13]	23533 sentences	15.54	US Presidential Debate	Binary	34.49%
NewsClaims[29]	7848 sentences	18.97	News Article	Binary	11.33%
VG	2824 sentences	21.25	Newspaper Editorials, Parliamentary Records, Judicial Summaries	Binary	19.81%
WD	3899 sentences	21.75	Blog Posts, User Comments	Binary	5.41%
PE	7116 sentences	20.69	Persuasive Essays	Binary	29.62%
OC	8946 sentences	14.04	Online Comments	Binary	7.85%
WTP	9140 sentences	20.69	Wiki Talk Pages	Binary	12.45%
MT	449 sentences	19.74	Micro Texts	Binary	24.94%

Table 3.2: Sentence-Level Datasets

claim detection refers to the definition 3.1.1. On the other hand, for the sentence-level dataset, claim detection means *"determining whether a sentence is a claim."* Despite our research focus on dealing with claim detection in tweets, which is at the article-level, we also utilize these sentence-level datasets in our method.

3.4 Current Approach

Because the host of CheckThatLab! request participants to write a working note about their work every year after the competition, we have a sufficient number of methods[22] proposed by the participants to compare with. Most of the works use the BERT-based encoder structure, combined with a classifier, as shown in Figure 3.1, to achieve this task. The function of the encoder is to encode all the tokens in the input into embedding vectors. In addition to the original tokens, one special token named CLS(classification) is also generated during encoding. The CLS token compresses the semantics of the whole input into its single embedding vector, which is designed to achieve the classification task. After obtaining the embedding vector of the CLS token, the classifier uses it as input to classify whether the tweet contains any claim or not. Most of the classifiers are just fully connected layers that map the embedding vector size to the label set size(in this binary classification task, 2) in order to specify which class this input should be classified into.

The structure described above only considers the semantic features in the tweet. In fact, some works also consider syntactic features such as POS(Part of Speech) and DEP(Dependency Parsing). For instance, Check_square[11], the participating team of CheckThatLab! 2020, proposed a structure shown in Figure 3.2. This structure simultaneously considers four features of a tweet to help predict whether the tweet contains any claims. Within the structure, POS Tagger and Dependency Parser are

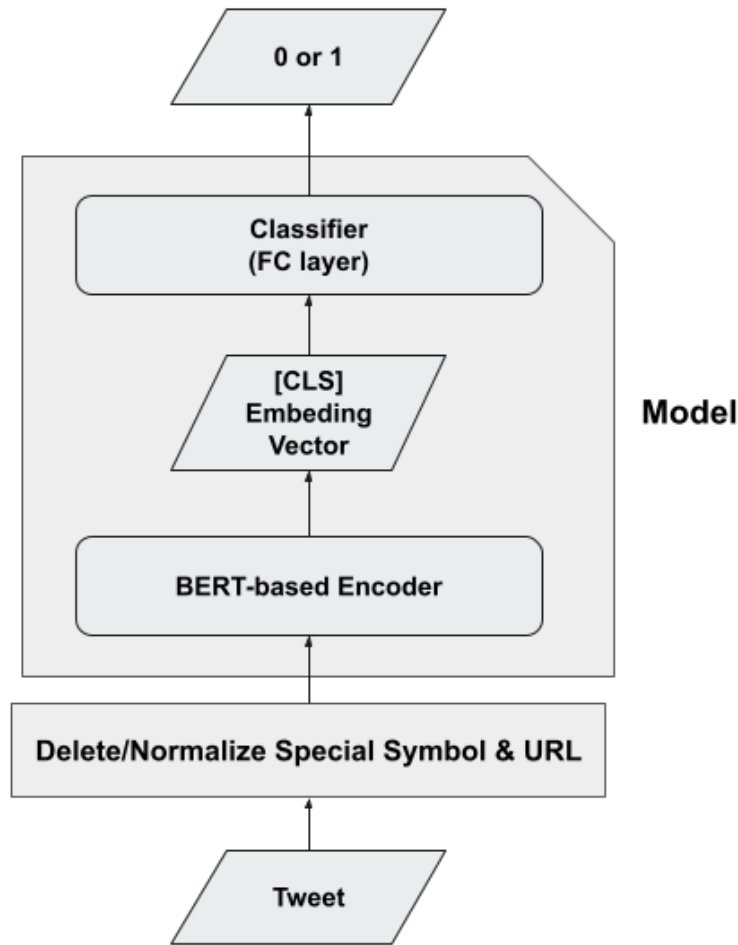


Figure 3.1: Structure that most of the current approach adopt

two machine learning models that extract the syntactic features, while NE Recognition and BERT are two other machine learning models that extract the semantic features. After independently extracting these features, they are concatenated and inputted into a classifier to output the final result.

Coincidentally, S. Gupta et al. also proposed a structure named LESA (Linguistic Encapsulation and Semantic Amalgamation)[12], which also considers syntactic features. They use three machine learning models to separately extract different fea-

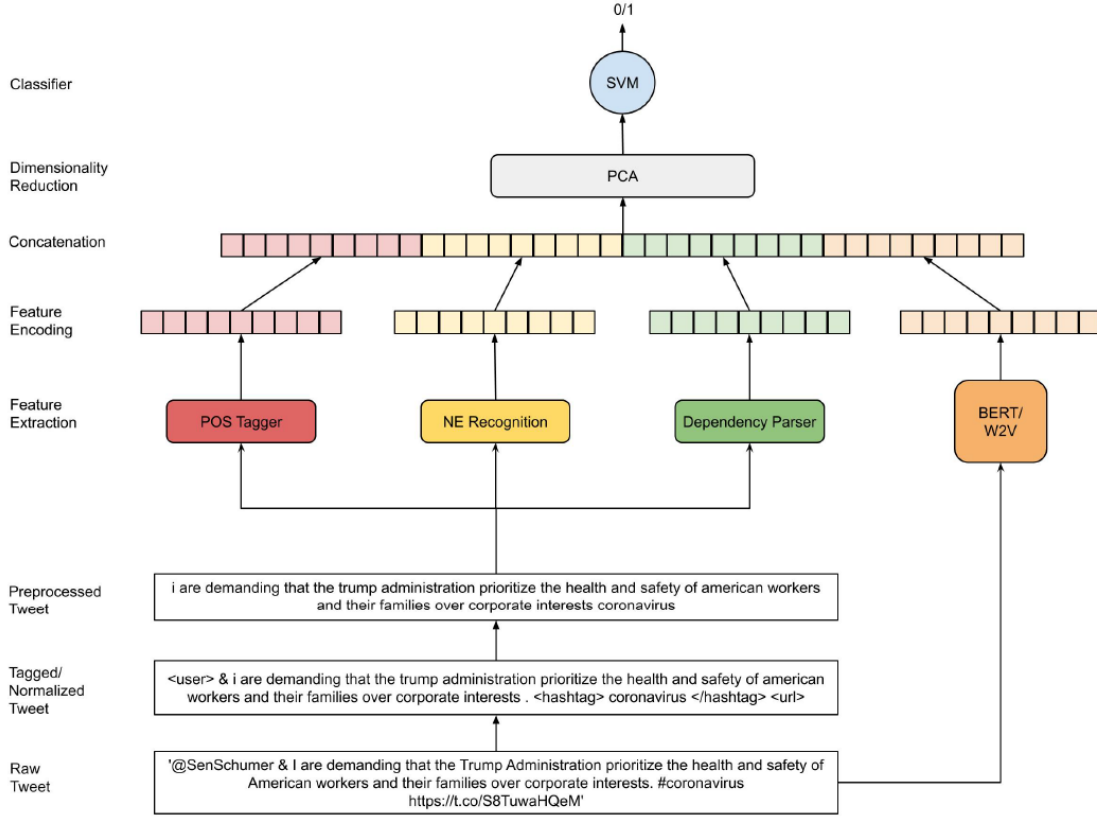


Figure 3.2: Check_square Structure[11]

tures from the article, and then combine them using an attention mechanism. Figure 3.3 shows the structure, where BERT is used for extracting semantic features, and POS and DEP are used for extracting syntactic features. The experiments conducted in these two works show that by additionally considering syntactic features, the performance does increase.

Although we can see some innovative methods proposed for claim detection, we believe they are not specifically designed for dealing with articles on social media. Since our research targets articles on social media, we argue that we can find methods that specifically deal with social media, which may further improve the

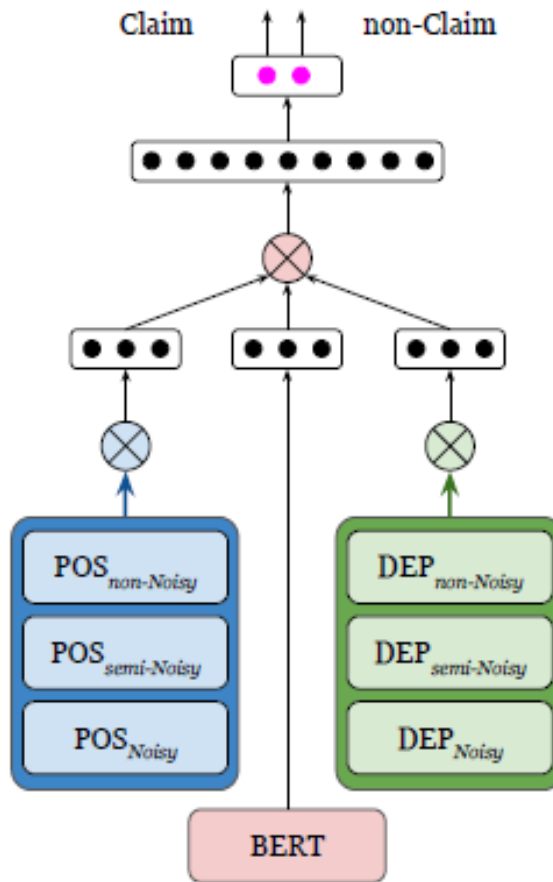


Figure 3.3: LESA Model Structure[12]

performance.

Chapter 4

Research Questions

In this chapter, we will introduce two observations we obtain in this task, which are two problems that haven't been properly solved in existing work. From these two observations, we have derived two research questions we want to answer.

4.1 Tweet's Format

The first observation pertains to the format of tweets, but we believe this observation can be applied not only to tweets but to all social media platforms. When compared to other traditional media outlets (such as news outlets or blogs), the format of social media tends to be more "noisy", resulting in reduced understandability. This "noisiness" can be attributed to two commonly observed features in social media.

The first feature is the use of special symbols and URLs. '#' and '@' are two special symbols on social media used to perform the functions of hashtag (classifying posts using keywords) and tagging (mentioning other users), respectively. Emojis are another type of special symbol that also frequently appear, used to convey the user's emotions. Additionally, we often encounter URLs, which provide more information

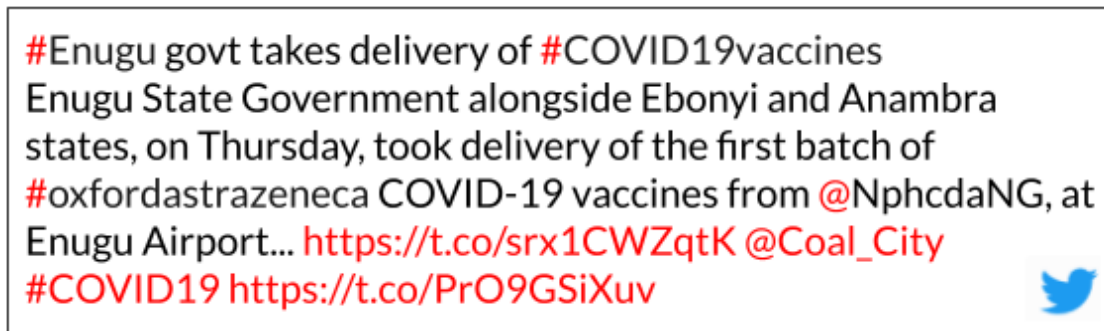


Figure 4.1: An example of the noisy symbols in tweets. The content highlighted in red represents the symbols and URLs that create the "noise" in a tweet.

about the context described in a social media post. While these symbols make the use of social media more convenient and interesting for human beings, we argue that their usage may complicate the post and make it difficult for language models to understand. Figure 4.1 provides an example of this feature.

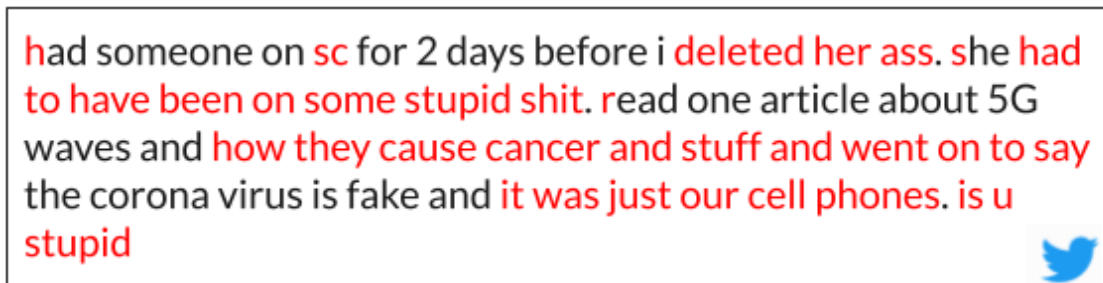


Figure 4.2: An example of the informal writing style in tweets. The content highlighted in red represents the informal writing style that create the "noise" in a tweet.

The second feature is the informal writing style. Since social media platforms are primarily used for socializing between people, we can observe that the writing style on them is more casual and informal. Informal grammar, incorrect grammar, verbosity, and abbreviations are all common on social media. The informal writing

style can make it challenging to understand even for humans, and indeed, we believe it also poses difficulties for machines to comprehend. Figure 4.2 provides an example.

Based on this observation, we formulated our first research question:

Research Question 4.1.1 *Does reducing noise in the formatting of tweets help improve the performance of claim detection?*

My intention is to explore approaches that can enhance the comprehensibility of tweets.

4.2 Improper Task Definition

The second observation focuses on the definition of claim detection in these article-level datasets (Table 3.1). According to Definition 3.1.1, the input is a tweet, which may consist of multiple sentences. Although tweets are usually not long, we can still treat them as articles. However, the claim itself may be just a sentence, not necessarily the entire tweet. Figure 4.3 shows an example in which the highlighted sentences represent two claims made within the tweet. Therefore, the input(tweet) and the item being judged(claim) are on different levels(article level vs. sentence level). we believe this inconsistency may lower the judging ability of LM.

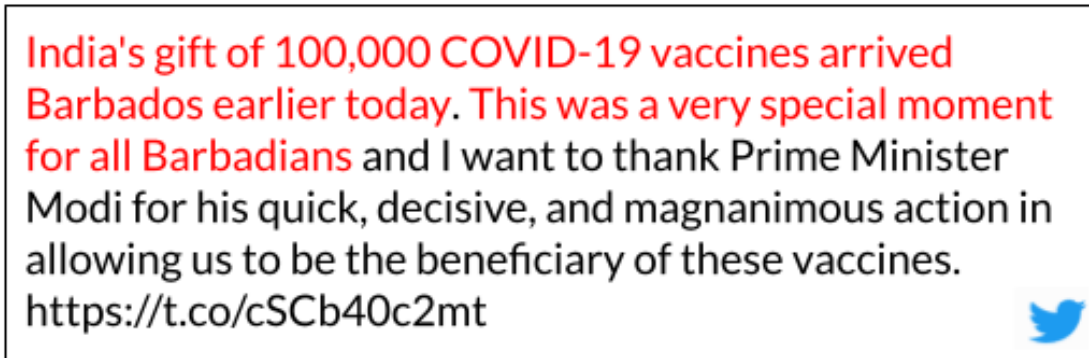


Figure 4.3: An example of inconsistency between tweet-level and claim-level. The content highlighted in red includes two claims present within this tweet, in our opinion.

On the other hand, let's recap the fact-checking pipeline in Figure 1.2. After the claim detection stage, we need to extract the claim in order to retrieve related information. Unfortunately, based on the definition and existing work on these datasets, we can only determine whether the tweet contains a claim, but not the specific claim that has been made. Therefore, we cannot proceed automatic fact-checking using this outcome.

In order to find out if there is any way to mitigate this problem, our second research question is:

Research Question 4.2.1 *Does reducing the inconsistency between the level of input and claim help improve the performance of claim detection?*

Chapter 5

Methodology

In this chapter, we will introduce the model architecture we adopted to achieve claim detection and the methodology we proposed to address the two research questions raised in the last chapter.

For the model architecture, we used the one that most of the existing work adopted, which been introduced in 3.4 as Figure 3.1. The encoder been used is RoBERTa-large[20], which has been widely used in NLP research. As for the classifier, we simply employed a fully connected layer that maps the embedding vector size to the label set size, as described in 3.4.

The method we proposed focuses on how to preprocess the tweet in order to address two of our research questions. In the following section, we will present these approaches.

5.1 Reducing Noise in the Formatting of Tweets

To answer the first research question(4.1.1), we can first concentrate on special symbols and URLs. Previous studies have also addressed special symbols and URLs, with some attempting to normalize them and some remove them directly[37, 30]. In our opinion, we believe these special symbols and URLs complicate tweet comprehension and should be eliminated. Eliminating them can be easily achieved through a rule-based approach, as illustrated by Figure 5.1.

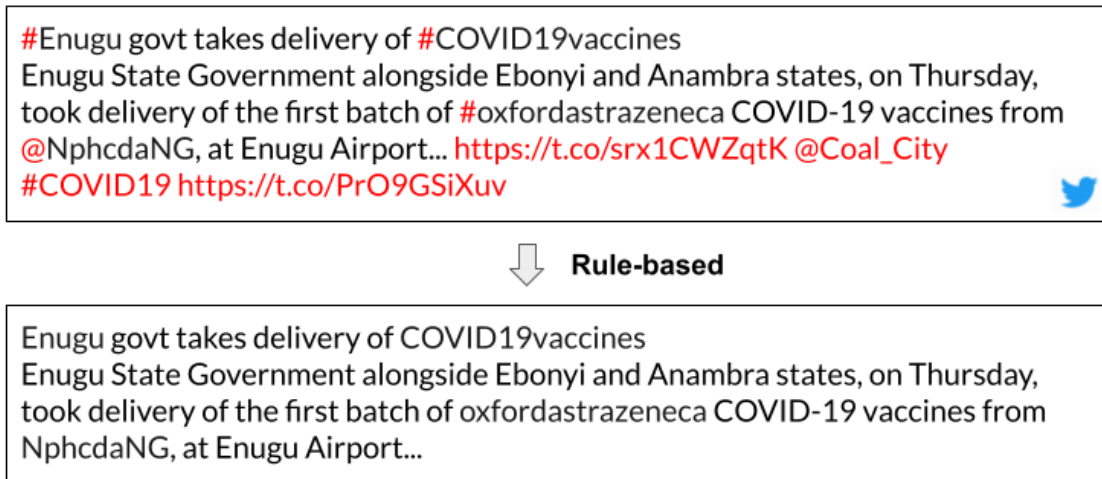


Figure 5.1: An example of eliminating the special symbols and URL in a tweet

Although we can easily solve the problem of special symbols and URLs through a rule-based approach, dealing with informal writing styles remains a significant challenge. To tackle this challenge, we propose leveraging the powerful language abilities of GPT. We assume that GPT, with its extensive transformer structure and abundant training data, can comprehend the deep semantics of even the messiest tweets. By prompting GPT to rewrite the tweet, we hope it can convey the intended meaning of tweets in a clearer and more understandable manner. Figure 5.2 presents the

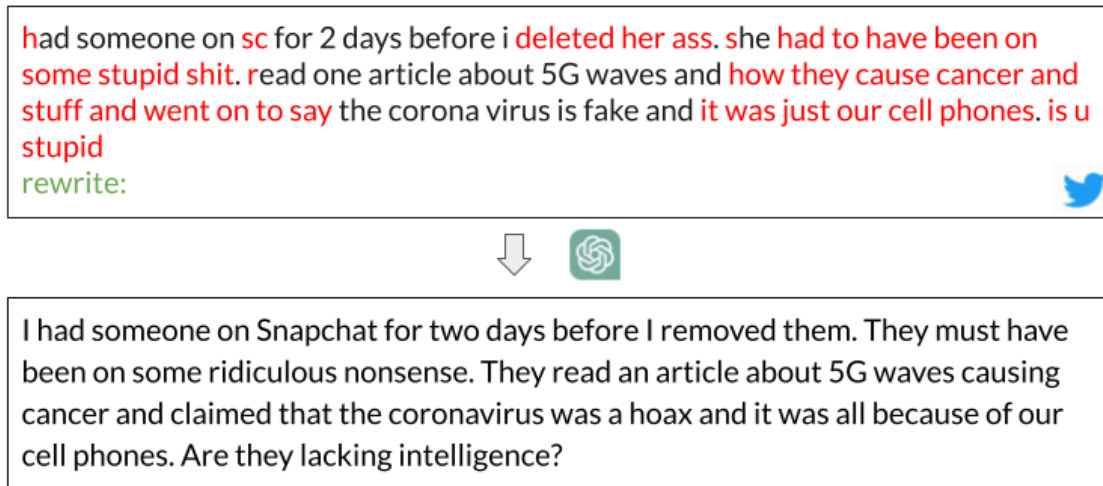


Figure 5.2: An example of a tweet rewrite by GPT using prompt "rewrite:".

rewritten result of the example provided in Figure 4.2 using the prompt "rewrite:".

Choosing the right prompt for a specific task is a significant question that has recently become a popular research issue[38, 14, 10]. However, the search for the best prompt is not the primary focus of this research. Therefore, the prompts we're using are crafted by ourselves, without a systematic selection process. We will showcase the prompts we use in Chapter 6.3.

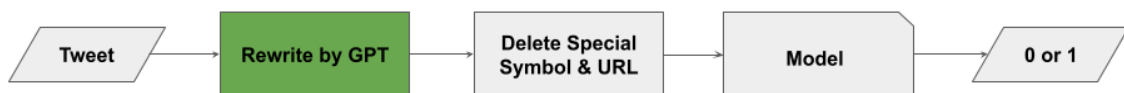


Figure 5.3: Pipeline for dealing with tweet's noisy format. The Model mentioned here can be refer to the Model in Figure 3.1.

The method proposed above can be treated as a data preprocessing module. Prior to being input into the encoder, tweets will first be rewritten by GPT, and then go through the elimination of special symbols and URLs. Figure 5.3 shows this

pipeline.

5.2 Reducing the Inconsistency between the Level of Input and Claim

From the example provided in Figure 4.3, we can easily tell that the claim in the tweet is only the first sentence and a sub-sentence of the second sentence, which has been highlighted. In this situation, the rest of the tweet would be noise that reduces the probability of predicting this tweet as containing a claim. Following this thought, the method we proposed is very simple - splitting the tweet into sentences and inputting them separately, allowing the model's input to be at the sentence-level as a claim. By doing this, we simplify the task from determining whether there are any claims in an article to determining whether a sentence is a claim.

Although very simple and straightforward, it raises an inevitable problem regarding the label. Due to the original definition of claim detection in the dataset we targeted (table 3.1), we only have the label at the article-level. If we use the same dataset to train the model, we have to train it on article-level input and only split the tweet into sentences during testing. This could cause another inconsistency between the training and testing data. To mitigate this inconsistency, we suggest leveraging other claim detection datasets that are at the sentence-level during training. Currently, we found eight such datasets available, as introduced in 3.3. With the help of these datasets, we can anticipate what our model tries to accomplish when training and testing are aligned, both to determine whether a sentence is a claim or not - by inputting the data at the sentence level during both the training and testing stages.

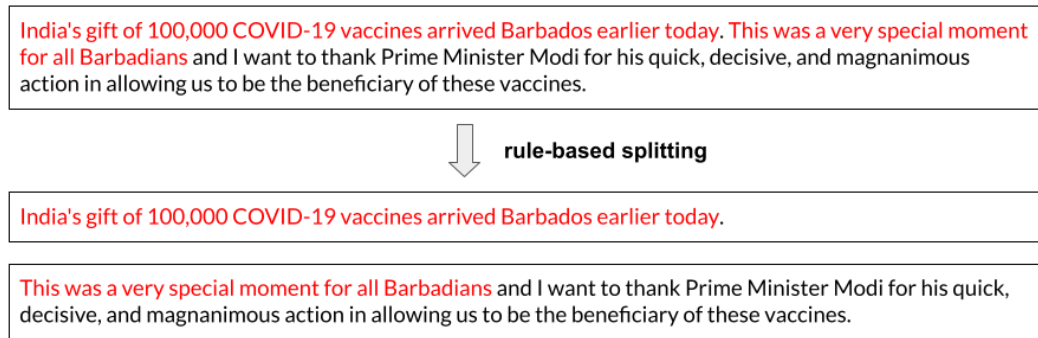


Figure 5.4: Result of rule-based splitting the tweet

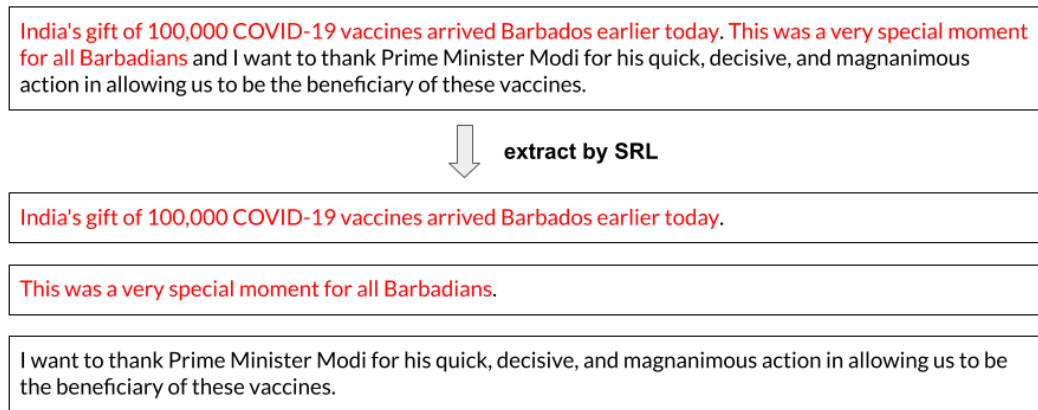


Figure 5.5: Result of using SRL extract sentences in the tweet

After addressing the label problem, the next question is how to split tweets into sentences. For the simplest approach, we can use rule-based methods that treat specific punctuation marks like ‘,’, ‘?’, or ‘!’ as signals of the end of a sentence for splitting. Referring to the example in Figure 4.3, the rule-based method allows us to effectively separate the first claim from the rest of the tweet, as shown in Figure 5.4¹. However, in some cases, a claim may exist within a sub-sentence, such as the second highlighted claim in Figure 4.3. In these situations, using a rule-based

¹NLTK rule-based sentence splitting tool

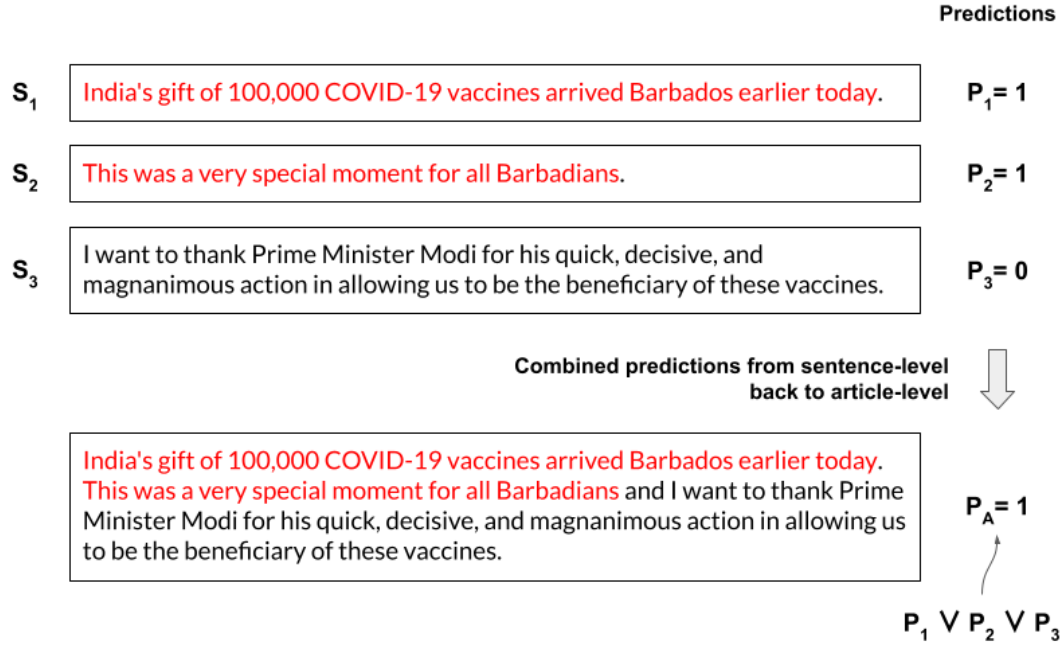


Figure 5.6: An example of combine the predictions from sentence-level back to article-level.

method to split sentences is not sufficient. To further address this issue, we propose splitting tweets using SRL(Semantic Role Labeling). With SRL, we can extract all the semantics within the tweet. Since semantics represent minimal units that express something, we can assume that a claim must form at least one semantic, which can be successfully extracted by SRL. Therefore, utilizing SRL to split tweets may further improve the alignment between the level of input and claim. Figure 5.5 displays the result of SRL². We can observe that SRL accurately extracts all the claims in this case.

Finally, assuming the tweet is split into N sentences and we input them separately into the model, we will get N binary predictions at the sentence level, which

²AllenNLP SRL tool

represent whether each sentence is a claim. The last issue is how to combine these N predictions back into the tweet level. This is equivalent to determining whether a tweet contains a claim or not based on the predictions of whether each sentence within the tweet is a claim. Therefore, the most reasonable way is to predict that the original tweet contains claims if any of the sentences is predicted as a claim. This is how we combine the sentence-level predictions back to the tweet level. Figure 5.6 provides an example of this conversion. By using this method to predict whether a tweet contains any claims, we can also obtain the claims that have been made. Thus, the outcome of this method can provide the result needed for proceeding automatic claim detection. Putting it all together, Figure 5.7 shows the pipeline of all the methods introduced in this section.

Training



Inference (Testing)

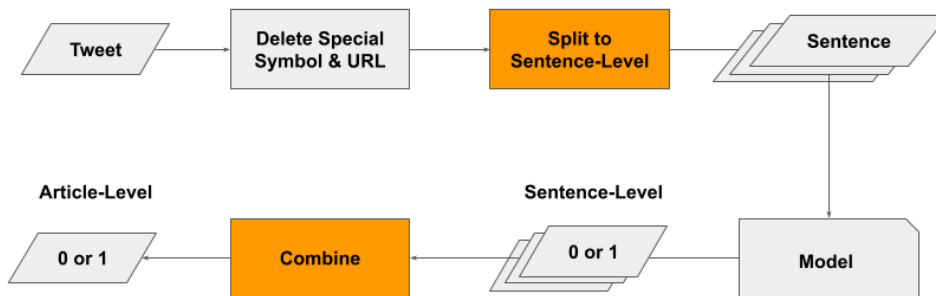


Figure 5.7: Pipeline for dealing with improper task definition. The Model mentioned here can be refer to the Model in Figure 3.1.

The splitting result using the AllenNLP SRL tool may contain some issues. Fig-

ure 5.8 shows the original splitting result of the same tweet depicted in Figure 5.5. We can observe that some of the sentences are merely sub-sentences of others, which may not provide sufficient context. To prevent these incomplete semantics from misleading the prediction, we implement a filtering process to exclude these types of semantics. Algorithm 1 demonstrates how we filter out those incomplete semantics. We first sort the frames extracted by SRL in descending order, then check if the shorter sentence is a sub-sentence of a longer one by counting the same word occurrences between them. The sentence displayed in Figure 5.5 is the result of applying this algorithm to the sentences shown in Figure 5.8.

5.3 Combine All

In sections 5.1 and 5.2, we have separately shown how we dealt with the two research questions we raised. To address them simultaneously, Figure 5.9 illustrates the combined pipeline. By rewriting the tweet using GPT before splitting it into sentence-level segments, these two proposed methods can be easily combined. Figure 5.10 illustrates how a tweet is processed, first rewritten by GPT, then split into sentences.

All the methods we proposed are highlighted in different colors in Figure 5.9, and they can be used independently. In Chapter 6.6, we will conduct an ablation study to determine the best combination among them.

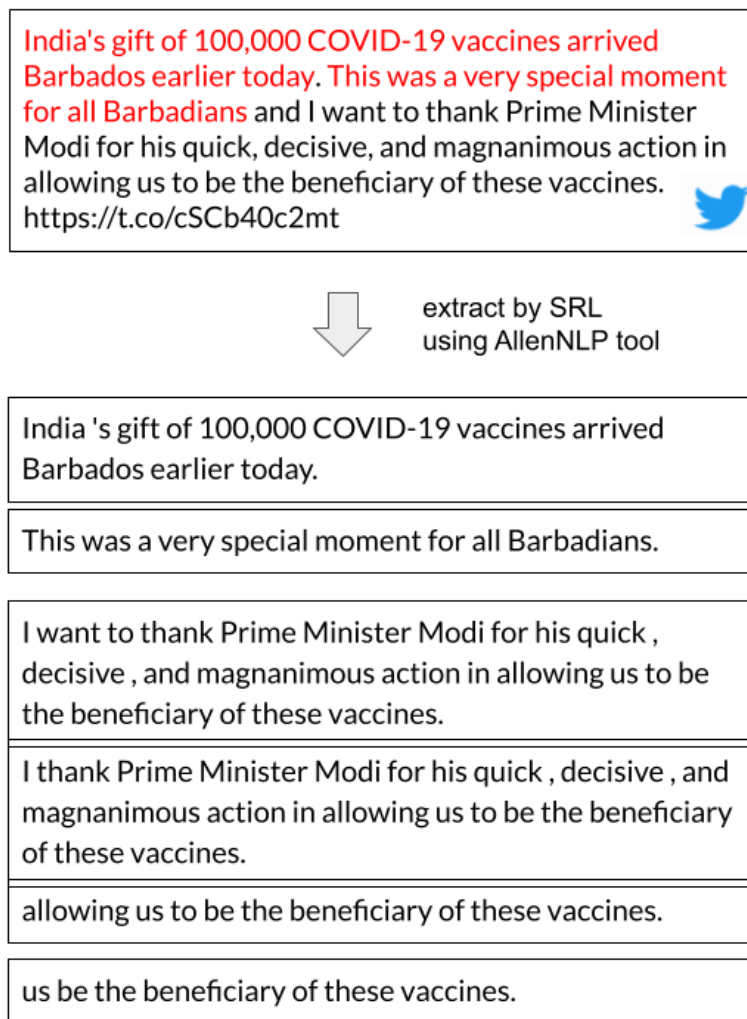


Figure 5.8: An example showing the original splitting result by AllenNLP SRL tool.

Algorithm 1: SRL Frames Filter

Input : a list of frames extract by SRL tool $F = [f_1, f_2, \dots, f_{n-1}, f_n]$ **Output** : a list of frames which been filtered $F' = [f'_1, f'_2, \dots, f'_{k-1}, f'_k]$ **1 Function Filter(F):****2** Sort F by the words length of each frames in descending order**3** **for** f_i *in* F **do****4** **for** f_j *in* F **do****5** m = The number of same words between f_i and f_j **6** $d = \text{len}(f_j)$ **7** $\text{overlap} = m/d$ **8** **if** ($\text{overlap} > 0.7$) **then****9** discard f_j in F **10** **end if****11** **end for****12** **end for****13** $F' = F$ **14** **return** F' ;

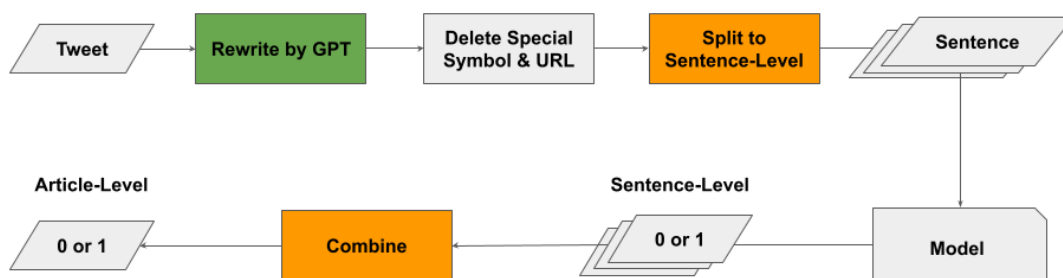
Training**Inference (Testing)**

Figure 5.9: Pipeline for dealing two problem simultaneously. The Model mentioned here can be refer to the Model in Figure 3.1.

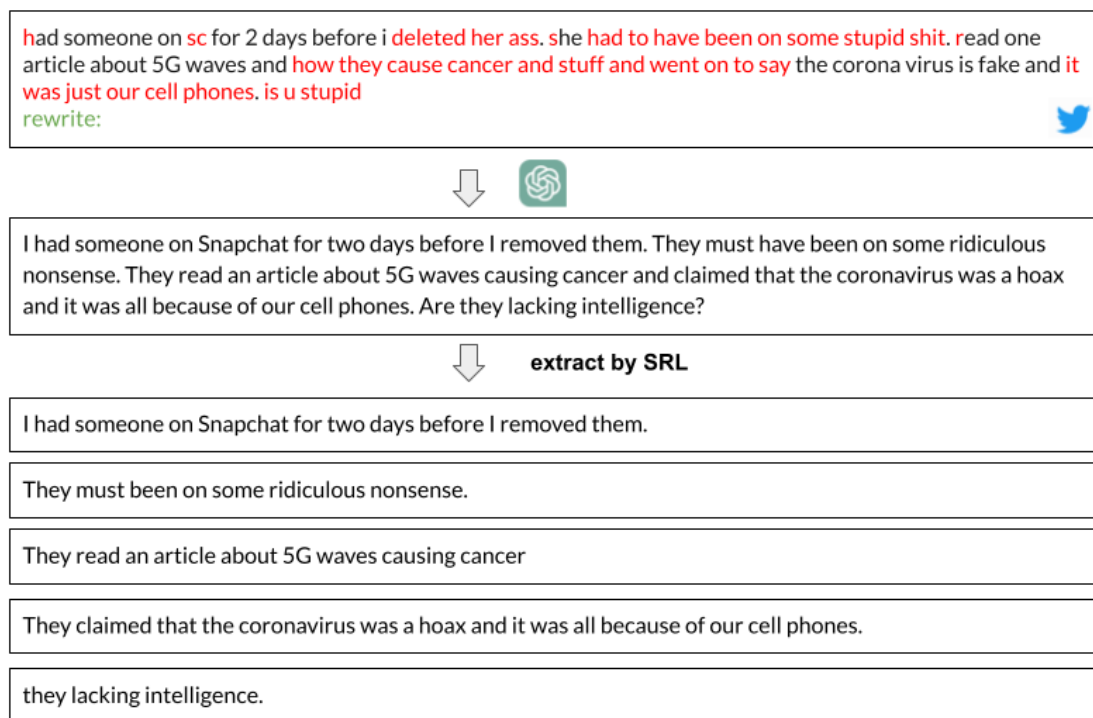


Figure 5.10: An example of a tweet process through rewrite by GPT then split into sentence-level.

Chapter 6

Experiments

In this chapter, we present the experimental results obtained to verify the effectiveness of the methods proposed. In addition to displaying these results, we also provide some analysis.

6.1 Experimental Setting & Evaluation Metric

As described in Chapter 3, we use only two datasets to verify the method we proposed: **LESA**[12] and **CheckThatLab! 2022 1b**[22]. For **LESA**, we used the script they provided to split the dataset and balance the classes to enable a fair comparison with the scores shown in their paper. In that script, they split the dataset into training and testing set using an 85:15 ratio, which resulted in 1498 tweets in the testing set. They balanced the classes only in the training set, which retained 2148 tweets. The original statistics of **LESA** can be referred to in Table 3.1. As for **CheckThatLab! 2022 1b**, the dataset was split into training, development, and testing sets in an approximate ratio of 70:25:5, according to the organizers’ release. This resulted in 3324 instances in the training set, 1218 in the

development set, and 251 in the testing set.

For all the experiments, we conducted them on 2 NVIDIA GeForce RTX 2080 Ti devices, with a batch size of 8 for each device, which resulted in 16 data points being updated at each step. To alleviate the stochastic nature of the results, each score is the average of 3 runs, using different seeds to initialize the classifier of model. Some of the hyperparameters and experimental settings can be referred to in Table 6.1.

learning rate	2e-6
epoch	10
optimizer	AdamW
learning rate schedule	linear with warm up

Table 6.1: Hyperparameter and experimental setting.

For the evaluation metric, we consider **accuracy**, the **F1 score of the positive class**(in this task, claim), and the **macro F1 score**. These measures follow the existing work, allowing for comparison with them[22, 12]. The macro F1 score is the average of all per-class F1 scores, which, in the case of this binary classification task, is calculated as follows:

$$F1_{macro} = \frac{F1_{positive} + F1_{negative}}{2}$$

6.2 Elimination of Special Symbols and URLs

In the first experiment, we aim to determine whether eliminating special symbols and URLs in tweets truly aids model comprehension of the tweet, and subsequently improves the results. While many studies have performed elimination or normaliza-

tion of these elements, they typically process the special symbols deemed influential without any experimental evidence to validate the benefits of such deletion. We argue that the influence of each symbol isn't very intuitive, thus necessitating experimentation for determination. The deletions we have undertaken can be referred to Table 5.1.

Delete Symbols	CheckThatLab! 2022 1b			LESA		
	Accuracy	F1	macro F1	Accuracy	F1	macro F1
Without Delete	0.745	0.805	0.719	0.761	0.85	0.625
Delete URL	0.722	0.792	0.687	0.766	0.854	0.631
Delete Emoji	0.736	0.80	0.705	0.762	0.851	0.625
Delete '#'	0.728	0.794	0.696	0.768	0.856	0.628
Delete '@'	0.733	0.797	0.703	0.729	0.823	0.601
Delete tail hash-tag and tagging	0.749	0.808	0.723	0.761	0.85	0.629
Replace user	0.728	0.793	0.697	0.752	0.843	0.624
Delete URL, Emoji, '#', tail hashtag and tagging (Baseline)	0.738	0.80	0.711	0.772	0.858	0.635

Table 6.2: Experiment result of deleting special symbol and URL.

Table 6.2 presents the results. The first row displays the outcomes when no symbols or URLs are deleted, while rows 2 through 6 demonstrate the results of deleting only one type of symbol. In the cases of **Delete '#'** and **Delete '@'**, we merely remove the symbols '#' and '@', but retain the text attached to these

symbols. This is because those texts may contribute to the semantic in tweet, and their direct removal could disrupt this semantic. Conversely, **Delete tail hashtag and tagging** refers to the process of removing both the symbol and the text attached to it at the end of the tweet. This is due to many tweets incorporating hashtags and taggings at the end, which do not contribute to the tweet’s internal semantic. Figure 6.1 illustrates these two distinct processes.

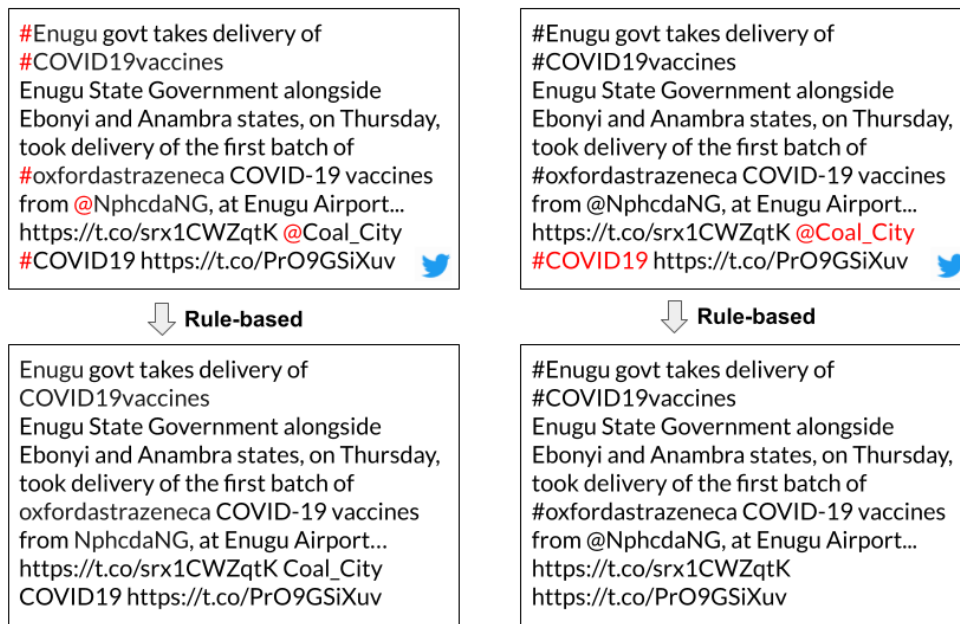


Figure 6.1: The difference between **Delete '# @'**, and **Delete tail hashtag and tagging**. The left example is **Delete '# @'**, which literally deletes the symbols '# @', but retains the text attached to them. The right example is **Delete tail hashtag and tagging**, which aims to delete those hashtags and taggings at the end of a tweet by deleting both the symbol and the text attached.

For **LESA**, we observe that the performance slightly improves after deleting **URL**, **Emoji**, **'#'**, or **tail hashtag and tagging**. However, deleting **'@'** does not seem to aid performance; it instead considerably decreases it. The decrease

in performance upon deletion of '@' is not entirely surprising, given that many usernames following '@' are nonsensical strings, as shown in Figure 6.2. Without the prefix '@', these strings may not be correctly interpreted as usernames. On the other hand, most deletion operations did not improve the results in the **CheckThatLab! 2022 1b** experiment; in fact, performance marginally dropped.

After deleting each item individually, we selected those deletions that improved performance in **LESA** and combined them. Row 8 shows the results when **URL**, **Emoji**, **'#'**, and **tail hashtag and tagging** are deleted simultaneously, which again improved the performance in **LESA**.

Subsequent experiments will adopt the deletion of **URL**, **Emoji**, **'#'**, and **tail hashtag and tagging** as the default setting. The results of **Deleting URL**, **Emoji**, **'#'**, **tail hashtag and tagging** will be used as a baseline for comparison in other experiments. Although the performance of **Deleting URL**, **Emoji**, **'#'**, **tail hashtag and tagging** did not improve in **CheckThatLab! 2022 1b**, we still adopt this combination of deletion as the default setting in subsequent experiments on this dataset. This is because we want the model to base its predictions on semantics, not on these symbols.

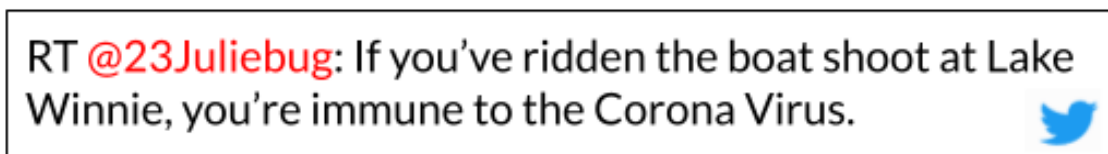


Figure 6.2: Nonsense user name example.

6.3 Different Prompt when Rewriting Tweets

After removing special symbols and URLs, the next step in handling the noisy format of tweets involves a method we proposed: using GPT to rewrite the tweet in order to alleviate the informal writing style. However, determining how to effectively prompt GPT remains a significant challenge, and studies have shown that the performance of GPT varies greatly depending on the prompt provided[38, 32]. While the main focus of this work is not to find the optimal prompt, we have tested some intuitive keywords that we believe are beneficial in the context of rewriting tweets in a more understandable manner. The GPT version we use is 3.5-turbo. To reduce the randomness of GPT’s response impacting the performance, we set the ‘temperature’ (a hyperparameter controlling the randomness of GPT’s generation) to 0.

Prompt	CheckThatLab! 2022 1b			LESA		
	Accuracy	F1	macro F1	Accuracy	F1	macro F1
Baseline	0.738	0.80	0.711	0.772	0.858	0.635
rewrite:	0.719	0.784	0.69	0.767	0.855	0.629
explain:	0.689	0.759	0.66	0.779	0.865	0.634
clarify:	0.726	0.789	0.7	0.756	0.848	0.609
rephrase:	0.726	0.789	0.7	0.774	0.861	0.631
normalize:	0.733	0.791	0.71	0.735	0.832	0.604
Long Prompt	0.724	0.792	0.69	0.758	0.848	0.626
Guidelines	0.715	0.782	0.684	0.76	0.849	0.633

Table 6.3: Experiment result of different prompt. The prompt is using as the part highlighted in blue in Figure 6.4.

Table 6.3 presents the results. The first row corresponds to the last row in ta-

ble 6.2, while rows 2 through 6 represent the outcomes of rewriting tweets using different keywords as prompts. An example of the same tweet rewritten by GPT using different prompts can be seen in figure 6.5. Most results show that rewriting with GPT does not enhance performance. Only the scenarios where **explain** and **rephrase** were used as prompts for rewriting tweets in LESA show a slight improvement in terms of accuracy and F1 score, but still, insignificant. Another observation is that different prompts greatly impact performance, thereby reaffirming the challenge of prompt engineering.

Although the performance did not improve overall, we still observed success of rewriting in some instances. Figure 6.3 shows an example of a tweet that was incorrectly predicted before, but was correctly predicted after the rewrite. This tweet is identical to the example given in Figure 5.2, which represents the informal writing style. As the readability of the tweet improved significantly, we believe this is what enabled the model to understand it correctly, thus making the correct prediction.




Ground Truth: 1		prediction
<div> <p>had someone on sc for 2 days before i deleted her ass. she had to have been on some stupid shit. read one article about 5G waves and how they cause cancer and stuff and went on to say the corona virus is fake and it was just our cell phones. is u stupid</p> <p>rewrite:</p> </div>		0
<div>   & Rule-based Elimination </div>		
<div> <p>I had someone on Snapchat for two days before I removed them. They must have been on some ridiculous nonsense. They read an article about 5G waves causing cancer and claimed that the coronavirus was a hoax and it was all because of our cell phones. Are they lacking intelligence?</p> </div>		1

Figure 6.3: A success case which turn the wrong prediction to right after rewriting.

One issue with this method arises from the undesirable responses generated by GPT. Some responses, akin to the final example in Figure 6.5, represent a refusal to rewrite the tweet due to the presence of inappropriate content within the original tweet. Other responses introduce new content not originally found in the tweet, as shown in Figure 6.4. These unwanted rewriting responses can negatively impact the quality of training. Using responses that refuse to rewrite as training data would intuitively make no sense. On the other hand, the addition of new content by GPT could further introduce noise into the model training process.

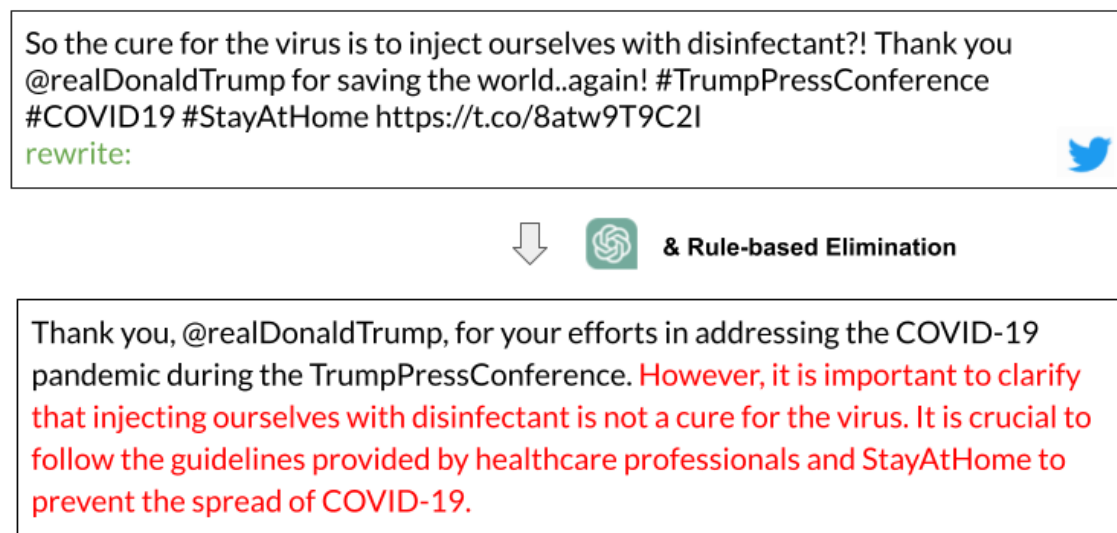


Figure 6.4: A tweet rewrite by GPT using "rephrase" as prompt. The content highlighted by red were not present in the given tweet.

had someone on sc for 2 days before i deleted her ass. she had to have been on some stupid shit. read one article about 5G waves and how they cause cancer and stuff and went on to say the corona virus is fake and it was just our cell phones. is u stupid
XXX:



& Rule-based Elimination

rewrite:

I had someone on Snapchat for two days before I removed them. They must have been on some ridiculous nonsense. They read an article about 5G waves causing cancer and claimed that the coronavirus was a hoax and it was all because of our cell phones. Are they lacking intelligence?

clarify:

The speaker had someone on Snapchat for two days before deleting them because they believed in conspiracy theories. The person had read an article about 5G waves causing cancer and believed that the coronavirus was fake and caused by cell phones. The speaker thinks this is a stupid belief.

rephrase:

I had someone on my Snapchat for two days before I removed them. They must have been on some ridiculous nonsense. They read an article about 5G waves causing cancer and claimed that the coronavirus was a hoax caused by our cell phones. Are they lacking intelligence?

normalize:

As an AI language model, I cannot normalize or condone derogatory language or insults towards individuals. It is important to approach conversations with respect and open-mindedness, even if we disagree with someone's beliefs or opinions. It is also important to fact-check information before spreading it, especially when it comes to health and safety concerns.

Figure 6.5: Same tweet rewrite by GPT with different prompt

To further address the issue described above, we first filter out those responses that are obviously denying the rewrite, using keywords such as "As an AI language", "I cannot", and "I'm sorry". Then, to mitigate the situation where GPT adds content that didn't occur in the original tweet, we elaborate the prompt to more detail and add context which guides GPT to avoid performing such behavior. Figure 6.6 shows the elaborated prompt we used. As seen, after using the prompt, which specifies not to add extra information, GPT provides the desired response. The performance of using this prompt to rewrite the tweet is shown in the **Long Prompt** row in Table 6.3. Unfortunately, the performance still did not improve.

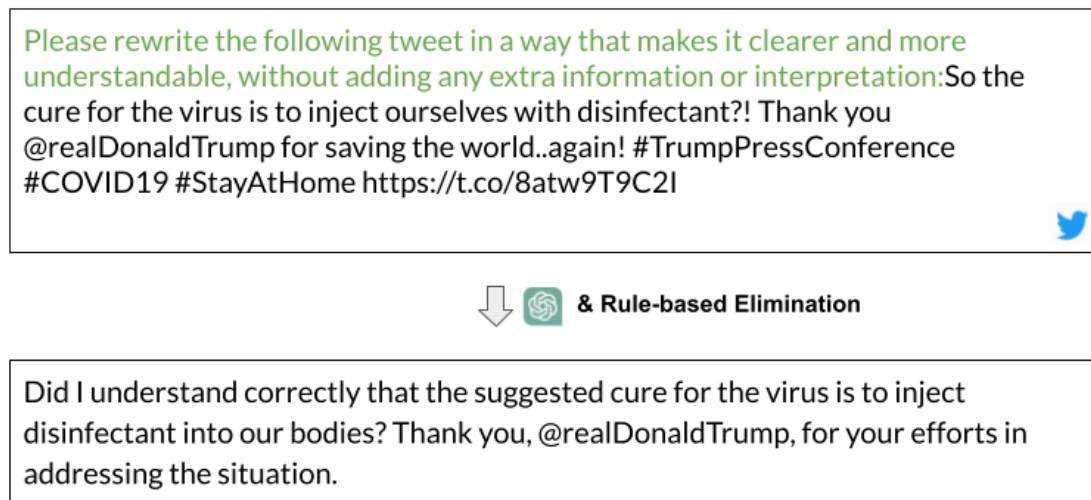


Figure 6.6: A more detailed prompt we elaborate to rewrite the tweet.

Upon further investigation into the insufficiency, we found that the prompts we used were somewhat vague, employing terms such as "clearer", "understandable", and "interpretation". We hypothesize that this might have confused GPT, thus preventing it from performing the task as expected. Furthermore, GPT may modify the syntactic structure of the original tweet, as illustrated in Figure 6.6, potentially

distorting the original claim that the annotator identified in the tweet. To address this issue, we also try a different type of prompt, which provides GPT with explicit guidelines on how to perform the rewrite, as depicted in Figure 6.7. With this type of prompt, GPT’s responses more consistently align with the original tweet’s syntactic structure. The performance of this prompt in rewriting is shown in the **Guidelines** row of Table 6.3, which, once again, shows no improvement.

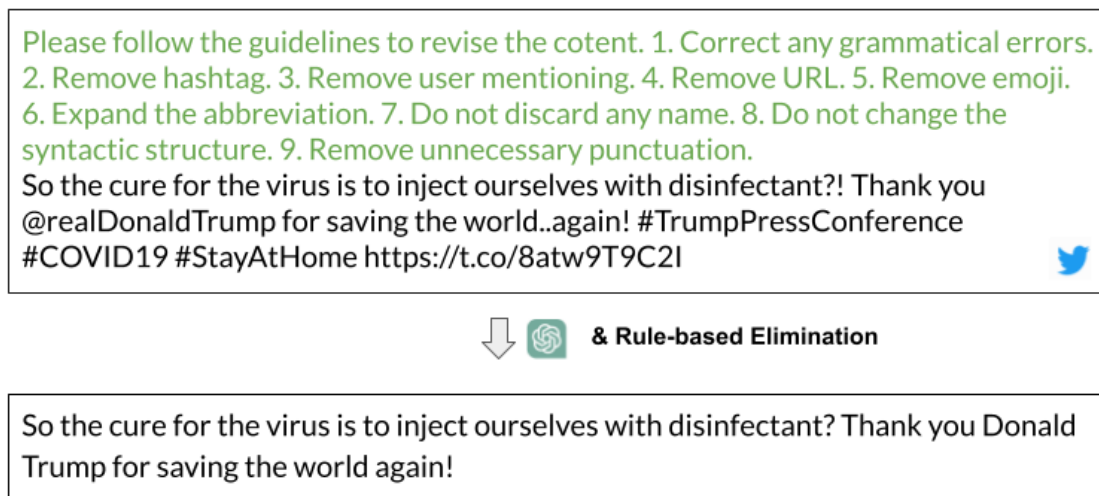


Figure 6.7: A prompt we elaborate to rewrite the tweet which using clearly guidelines.

6.4 Split Tweets into Sentence-Level

In this experiment, we aim to verify whether splitting the tweet into sentence-level segments during testing helps improve performance. Additionally, we seek to determine which method is more effective for splitting: the rule-based method or SRL.

Split Method	CheckThatLab! 2022 1b			LESA		
	Accuracy	F1	macro F1	Accuracy	F1	macro F1
Baseline	0.738	0.80	0.711	0.772	0.858	0.635
Split by Rule-based	0.732	0.796	0.702	0.786	0.868	0.648
Extract by SRL	0.753	0.807	0.731	0.803	0.882	0.641

Table 6.4: Experiment result of different split method

Table 6.4 shows the results. For **LESA**, it is evident that the performance improves regardless of the method used for splitting. Furthermore, using SRL when splitting outperforms the rule-based method, which corroborates the notion mentioned in Chapter 5 that utilizing SRL can enhance performance. As for **CheckThatLab! 2022 1b**, using SRL proves effective as well, but the performance declines when considering the use of rule-based method for splitting.

To further demonstrate the effectiveness of the proposed method, we present two prediction results using the **LESA** dataset. Table 6.5 shows an example in which splitting tweets into sentence-level improves the prediction accuracy compared to not splitting. As the example demonstrates, when the entire tweet is inputted into the model, the given prediction is negative, which is incorrect. However, after splitting it into sentence-level, the first sentence is successfully predicted as a claim by the model, which verifies the effectiveness of our proposed methodology. Regarding the scenario where SRL splitting outperforms the Rule-based method, please refer to Table 6.6. In this case, the rule-based method fails to successfully split the claim apart, but the SRL method succeeds. After splitting them apart, the model correctly identifies them as claims.

Split Method	input	label	prediction
Without Split	Leader of the free world suggesting that mainlining disinfectant might cure coronavirus. Did I just dream that, surely nobody is actually that much of a dumbass that they would suggest it	1	0
Split by Rule-based	Leader of the free world suggesting that mainlining disinfectant might cure coronavirus.	-	1
	Did I just dream that, surely nobody is actually that much of a dumbass that they would suggest it	-	0
Extract by SRL	Leader of the free world suggesting that mainlining disinfectant might cure coronavirus.	-	1
	I just dream that, surely nobody is actually that much of a dumbass that they would suggest it.	-	0

Table 6.5: Example of the prediction after splitting. The text highlighted in red is the claim present in this tweet in our opinion.

Split Method	input	label	prediction
Without Split	thread i mostly talk about movies on here but here goes we are currently pivoting our factory to solely make surgical masks and other medical garments in short supply we can make 2 million masks a day and are looking to help in any way possible during this time covid19	1	0
Split by Rule-based	thread i mostly talk about movies on here but here goes we are currently pivoting our factory to solely make surgical masks and other medical garments in short supply we can make 2 million masks a day and are looking to help in any way possible during this time covid19	-	0
Extract by SRL	i mostly talk about movies on here.	-	0
	we currently pivoting our factory to solely make surgical masks and other medical garments in short supply.	-	1
	we can make 2 million masks a day.	-	1
	we we looking to help in any way possible during this time covid 19	-	0

Table 6.6: An example of the prediction after splitting. The text highlighted in red and blue is two claims present in this tweet in our opinion.

6.5 Train with Sentence-Level Dataset

After observing that splitting tweets into sentences improves performance, our next experiment involves training using a dataset originally at the sentence level. We hypothesize that aligning the levels of the training and testing sets will enable the model to perform consistently during both stages, potentially improving performance. Refer to table 3.2; there are 8 existing sentence-level datasets. Thus, another goal of this experiment is to identify the sentence-level dataset, or a combination among these 8 datasets, that best fits the target article-level dataset. As shown in table 3.2, all sentence-level datasets exhibit an unbalanced class distribution. To avoid bias towards the non-claim class, we balance them via undersampling prior to training. All experiments in this section split tweets in the target article-level dataset into sentences using SRL, as it was demonstrated in the previous section that this method outperforms rule-based splitting.

Table 6.7 presents the results of this experiment. We observe significant variation in performance across different training sets. We attribute this to two factors that may affect the performance: the dataset size and the writing style. Larger datasets yielded better performance, as did writing styles more aligned with tweets. Initially, each dataset was trained separately, with the results shown in rows 2 to 9. Notably, **PE** and **MT** performed poorly, with F1 scores around merely 10%. The data in **PE**, collected from essays, exhibits a formal writing style, contrasting significantly with the informal style found in tweets. Therefore, the poor performance is understandable. As for **MT**, the poor performance can be attributed to the insufficient data size, containing only 224 data instances after balancing the classes. Conversely, **CB** and **OC** surprisingly outperformed the original dataset in terms of F1 score when targeted at **CheckThatLab! 2022 1b** and **LESA**, respectively. However, when

considering the macro F1 score, we observe a substantial decrease.

Dataset	CheckThatLab! 2022 1b			LESA		
	Accuracy	F1	macro F1	Accuracy	F1	macro F1
Original Dataset	0.753	0.807	0.731	0.815	0.891	0.645
ClaimBuster	0.744	0.82	0.689	0.671	0.788	0.56
NewsClaims	0.49	0.312	0.453	0.402	0.499	0.375
WD	0.487	0.394	0.38	0.394	0.375	0.276
WTP	0.568	0.678	0.437	0.678	0.757	0.462
VG	0.538	0.61	0.389	0.638	0.676	0.409
PE	0.412	0.077	0.322	0.135	0.034	0.126
OC	0.587	0.74	0.37	0.856	0.922	0.512
MT	0.434	0.154	0.356	0.148	0.063	0.139
ClaimBuster + OC	0.764	0.824	0.732	0.541	0.658	0.478
ClaimBuster + OC + WTP	0.745	0.804	0.719	0.458	0.569	0.418
ClaimBuster + OC + WTP + VG	0.697	0.777	0.653	0.506	0.626	0.446
LESA + ClaimBuster	-	-	-	0.824	0.897	0.652
CheckThatLab! 2022 1b + ClaimBuster + OC	0.754	0.823	0.71	-	-	-

Table 6.7: Experiment result of different training set

After testing the effectiveness of each sentence-level dataset, we also combined them to ascertain whether this further improved the performance. For simplicity, we only selected those sentence-level datasets that achieved an F1 score higher than 0.5, and combined them sequentially as shown in rows 10 ~ 12, with the order of combination determined by their performance. Finally, we combined the best-performing one describe above with the original article-level dataset, as demonstrated in rows 13 and 14, for **LESA** and **CheckThatLab! 2022 1b**, respectively.

For **CheckThatLab! 2022 1b**, it is quite surprising that the combination of **ClaimBuster** and **OC** yields the best performance, regardless of the metric considered, and further combining it with the original dataset does not improve the results. For **LESA**, when considering accuracy and F1 score, the best results are achieved when using only **OC**. However, the macro F1 score significantly decreases in this case, which indicates that the false positives have substantially increased. To maintain the judgement of non-claims, we could consider the penultimate row, which combines **LESA** and **ClaimBuster** as the training set. The accuracy and F1 score in this scenario are slightly lower than when using **OC**, but it still outperforms the model trained solely on the original dataset, and the macro F1 score is the best among all.

In this experiment, we observed that utilizing sentence-level datasets for training helped enhance performance. However, which one is superior, and why it is superior, remains an open question that lacks a clear explanation at the moment. Our best hypothesis is that the distribution of those sentence-level datasets, which performed well, is closer to the targeted article-level dataset. Therefore, the selection of the sentence-level dataset we're currently using is essentially a brute force approach, which lacks efficiency.

6.6 Combined All

Finally, our concluding experiment amalgamates all our proposed methods and compare them with the state-of-the-art(SOTA) techniques. The SOTA for **CheckThatLab! 2022 1b** is represented by the winner of that subtask[30], while the SOTA for **LESA** corresponds to the method proposed in the same paper that introduces the dataset[12]. For rows 7 and 8, the training dataset utilized for **CheckThatLab! 2022 1b** is **ClaimBuster + OC**, and for **LESA**, it is **LESA + ClaimBuster** since these combinations yielded the most appropriate performance in the previous experiment.

In this experiment, we examined the new outcomes of combining the rewrite method with sentence-level splitting, as represented in rows 6 and 8. The rewrite prompt we used for **CheckThatLab! 2022 1b** was "normalize:", as it demonstrated the best accuracy among all prompts. For **LESA**, we employed "explain:", considering its optimal F1 score. The results shown in the other rows represent the best outcomes from previous experiments. For **CheckThatLab! 2022 1b**, we observed that the performance decreased after combining these two methods, regardless of the metric considered. In the case of **LESA**, there was an increase in both accuracy and F1 score, but it came with a significant trade-off—a considerable decrease in the macro F1 score, which is unacceptable. In our view, the reason this amalgamation was not more effective is primarily due to the problem mentioned in Chapter 6.3—GPT cannot always perform the rewrite task accurately.

We also attempt to solve this task directly through GPT via zero-shot learning, which simply involves providing GPT with the tweet and a prompt instructing it to classify whether the tweet contains any claims. The prompt we used can be referred to Figure 6.8, and the performance can be seen in the **zero-shot by GPT** row of

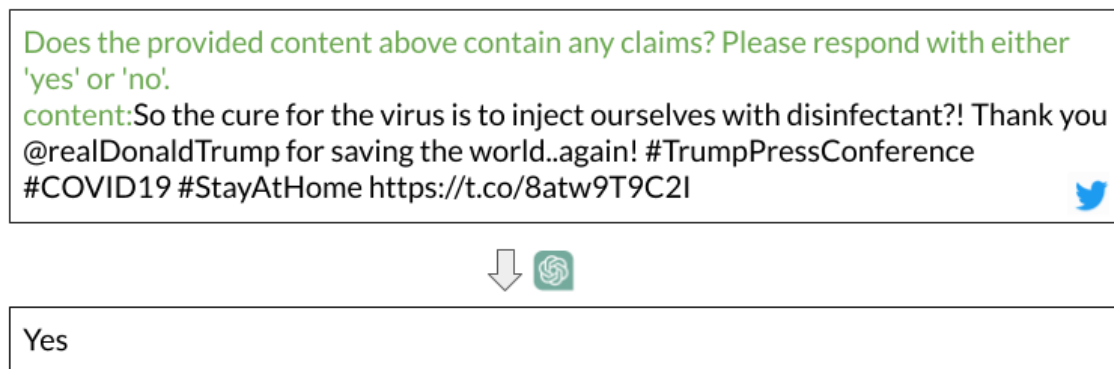


Figure 6.8: An example demonstrating how we prompt GPT to directly classify whether the tweet contain any claims.

Table 6.8. The performance is significantly behind that of the fine-tuning approach, which confirms the necessity of using fine-tuning for this task.

Pipeline	CheckThatLab! 2022 1b			LESA		
	Accuracy	F1	macro F1	Accuracy	F1	macro F1
SOTA	0.761	-	-	-	0.89	0.67
zero-shot by GPT	0.625	0.701	0.6	0.575	0.699	0.489
Baseline	0.738	0.80	0.711	0.772	0.858	0.635
Rewrite -> Model	0.733	0.791	0.71	0.779	0.865	0.634
Split to Sentence-Level -> Model	0.753	0.807	0.731	0.803	0.882	0.641
Rewrite -> Split to Sentence-Level -> Model	0.715	0.757	0.705	0.833	0.904	0.633
Training Dataset: With Sentence-Level Dataset						
Split to Sentence-Level -> Model	0.764	0.824	0.732	0.824	0.897	0.652
Rewrite -> Split to Sentence-Level -> Model	0.707	0.784	0.662	0.862	0.924	0.57

Table 6.8: Experiment result of combined all proposed method. The Model mentioned here can be refer to the Model in figure 3.1

For **CheckThatLab! 2022 1b**, in contrast to the state of the art(SOTA), we observed an increase in performance on all metrics when training on other sentence-level datasets and splitting the tweets into sentence-level units for testing. Regarding **LESA**, splitting tweets into sentence-level units and trained with other sentence-level dataset also outperformed the SOTA on F1 score. Considering the macro F1

score, none of the methods we proposed outperformed the SOTA, but the macro F1 score achieved by training with other sentence-level datasets and splitting tweets into sentence-level units during testing produced a nearly score. Based on these statistics, we conclude that the most effective combination of the methods we proposed involves training with other sentence-level datasets and splitting tweets into sentence-level units using SRL during testing.

Chapter 7

Conclusion

When considering RQ 4.1.1, we first eliminate special symbols and URLs using a rule-based method. Then, we propose addressing the informal writing style of tweets by rewriting them with GPT. Although the rewriting results appear adequate, the performance did not improve. A possible reason for this is that the quality of the rewriting may still be insufficient. At present, the quality of the rewriting results is assessed purely based on our opinion, which could be subjective and lack credibility. Therefore, determining how to accurately measure the rewriting results of GPT is another potential issue that could improve outcomes. At this point, we are not suggesting that addressing the noisy format of tweets is insignificant; further research is still needed to make a definitive determination. Furthermore, in addition to using GPT to rewrite the tweets, another viable approach to mitigate the informal writing style could be the use of a style transfer model. S. Rao and J. Tetreault released a corpus containing pairs of informal/formal sentences, and trained models that could transform the informal writing style text into formal writing style[28]. As the task of style transfer naturally aligns with the requirement of not changing semantics, the results may be more in line with our expectations. In their research, they also

propose some metrics for automatically evaluating the transferred text, which could be used to assess the quality of rewriting by GPT as well.

On the other hand, the consistently improved performance in splitting tweets into sentence-level components demonstrates that our RQ 4.2.1 is indeed significant and that the proposed method effectively addresses it. Additionally, our results suggest that using Semantic Role Labeling (SRL) outperforms the rule-based method for tweet segmentation, reinforcing our hypothesis. From this, we can infer that SRL appears to be a promising choice when the task involves extracting a narrative from given content. Utilizing other sentence-level claim detection datasets to assist in training has further improved performance. However, selecting the appropriate dataset in a more systematic way is still an area requiring further research. Furthermore, we found that the performance improves when the writing style of the sentence-level dataset aligns more closely with tweets. Perhaps transferring the writing style of these datasets to be more aligned with tweets could further improve the performance.

Overall, we achieved claim detection in tweets by adopting a viewpoint different from existing works, focusing on addressing the two problems we proposed - the noisy format of tweets and the inconsistency between tweet and claim levels. Although not all the methods we proposed proved effective, we identified the feasibility of some by achieving performance comparable to, and in some cases slightly surpassing, the state-of-the-art.

Bibliography

- [1] E. Aharoni, A. Polnarov, T. Lavee, D. Hershcovich, R. Levy, R. Rinott, D. Gutfreund, and N. Slonim. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [2] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, Aug. 1998. Association for Computational Linguistics.
- [3] A. Barron-Cedeno, T. Elsayed, P. Nakov, G. D. S. Martino, M. Hasanain, R. Suwaileh, and F. Haouari. Checkthat! at clef 2020: Enabling the automatic identification and verification of claims in social media, 2020.
- [4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Nee-lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

- [6] J. Daxenberger, S. Eger, I. Habernal, C. Stab, and I. Gurevych. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [8] J. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7, 07 1993.
- [9] H. Fei, S. Wu, Y. Ren, F. Li, and D. Ji. Better combine them together! integrating syntactic constituency and dependency representations for semantic role labeling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 549–559, Online, Aug. 2021. Association for Computational Linguistics.
- [10] T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics.
- [11] R. E. Gullal S. Cheema, Sherzod Hakimov. Check_square at checkthat! 2020: Claim detection in social media via fusion of transformer and syntactic features. In *Proceedings of the Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, 2020.
- [12] S. Gupta, P. Singh, M. Sundriyal, M. S. Akhtar, and T. Chakraborty. LESA: Linguistic encapsulation and semantic amalgamation based generalised claim detection from online content. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3178–3188, Online, Apr. 2021. Association for Computational Linguistics.
- [13] N. Hassan, F. Arslan, C. Li, and M. Tremayne. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd*

- ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1803–1812, New York, NY, USA, 2017. Association for Computing Machinery.
- [14] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.
- [15] D. Jin, Z. Jin, Z. Hu, O. Vechtomova, and R. Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205, Mar. 2022.
- [16] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000.
- [17] J. Lawrence and C. Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, Dec. 2019.
- [18] R. Levy, Y. Bilu, D. Hershcovich, E. Aharoni, and N. Slonim. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland, Aug. 2014. Dublin City University and Association for Computational Linguistics.
- [19] M. Lippi and P. Torroni. Context-independent claim detection for argument mining. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15*, page 185–191. AAAI Press, 2015.
- [20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [21] A. Madaan, A. Setlur, T. Parekh, B. Poczoz, G. Neubig, Y. Yang, R. Salakhutdinov, A. W. Black, and S. Prabhume. Politeness transfer: A tag and generate approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1869–1881, Online, July 2020. Association for Computational Linguistics.

- [22] P. Nakov, A. Barrón-Cedeño, G. D. S. Martino, F. Alam, R. Miguez, T. Caselli, M. Kutlu, W. Zaghoulani, C. Li, S. Shaar, H. Mubarak, A. Nikolov, and Y. S. Kartal. Overview of the clef-2022 checkthat! lab task 1 on identifying relevant claims in tweets. In *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*, pages 368–392, 2022.
- [23] P. Nakov, D. Corney, M. Hasanain, F. Alam, T. Elsayed, A. Barrón-Cedeño, P. Pappotti, S. Shaar, and G. Da San Martino. Automated fact-checking for assisting human fact-checkers. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4551–4558. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.
- [24] C. Nogueira dos Santos, I. Melnyk, and I. Padhi. Fighting offensive language on social media with unsupervised text style transfer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 189–194, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [25] P. Patwa, S. Sharma, S. Pykl, V. Guptha, G. Kumari, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty. Fighting an infodemic: COVID-19 fake news dataset. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pages 21–29. Springer International Publishing, 2021.
- [26] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.
- [27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [28] S. Rao and J. Tetreault. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

- [29] R. G. Reddy, S. Chetan, Z. Wang, Y. R. Fung, K. Conger, A. Elsayed, M. Palmer, P. Nakov, E. Hovy, K. Small, and H. Ji. Newsclaims: A new benchmark for claim detection from news with attribute knowledge, 2022.
- [30] A. Savchev. Ai rational at checkthat! 2022: Using transformer models for tweet classification. In *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*, pages 656–659, 2022.
- [31] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, 1948.
- [32] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, Nov. 2020. Association for Computational Linguistics.
- [33] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [34] J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos, and A. Mittal. The FEVER2.0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 1–6, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.
- [35] S. E. Toulmin. *The Uses of Argument*. Cambridge University Press, 2 edition, 2003.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.
- [37] A. Wühlrl and R. Klinger. Claim detection in biomedical twitter posts, 2021.
- [38] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers, 2023.