Factoring Fact-Checks: Structured Information Extraction from Fact-Checking Articles

Shan Jiang,[♠] Simon Baumgartner,[◇] Abe Ittycheriah,[◇] Cong Yu[◇]
[♠]Northeastern University, [◇]Google Research
sjiang@ccs.neu.edu,{simonba,aittycheriah,congyu}@google.com

ABSTRACT

Fact-checking, which investigates claims made in public to arrive at a verdict supported by evidence and logical reasoning, has long been a significant form of journalism to combat misinformation in the news ecosystem. Most of the fact-checks share common structured information (called *factors*) such as *claim*, *claimant*, and *verdict*. In recent years, the emergence of ClaimReview as the standard schema for annotating those factors within fact-checking articles has led to wide adoption of fact-checking features by online platforms (e.g., Google, Bing). However, annotating fact-checks is a tedious process for fact-checkers and distracts them from their core job of investigating claims. As a result, less than half of the fact-checkers worldwide have adopted ClaimReview as of mid-2019. In this paper, we propose the task of *factoring fact-checks* for automatically extracting structured information from fact-checking articles. Exploring a public dataset of fact-checks, we empirically show that factoring fact-checks is a challenging task, especially for fact-checkers that are under-represented in the existing dataset. We then formulate the task as a sequence tagging problem and fine-tune the pre-trained BERT models with a modification made from our observations to approach the problem. Through extensive experiments, we demonstrate the performance of our models for well-known fact-checkers and promising initial results for under-represented fact-checkers.*

CCS CONCEPTS

• **Information systems** → **Information extraction**; **Markup languages**; *World Wide Web*; *Structured text search*.

KEYWORDS

ClaimReview; fact-checking; misinformation; computational journalism; information extraction; sequence tagging; BERT

ACM Reference Format:
Shan Jiang,[♠] Simon Baumgartner,[◇] Abe Ittycheriah,[◇] Cong Yu[◇]. 2020. Factoring Fact-Checks: Structured Information Extraction from Fact-Checking Articles. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3366423.3380231

*Work done while the first author was an intern at Google Research.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380231

Fact-check: ... The Facebook post says that "D.A.R.E. removed cannabis from its list of gateway drugs." That claim seems to stem from ... We rate this Facebook post False.

Claim: "D.A.R.E. removed cannabis from its list of gateway drugs."
Claimant: Viral image
Verdict: False

Figure 1: Fact-check example and its three factors. This paragraph is excerpted from a fact-check published by PolitiFact on May 30th, 2019 titled "D.A.R.E. still thinks marijuana is a dangerous drug for kids". In this example, the claim "D.A.R.E removed..." and verdict "False" can be found in the text content of the fact-check, while the claimant "Viral image" is replaced by "Facebook post". These factors are reported by PolitiFact.

1 INTRODUCTION

As a means to combat misinformation [1, 17], journalists conduct research with evidence and logical reasoning to determine the veracity and correctness of factual claims made in public, and publish fact-checking articles (or *fact-checks*) on their news outlets [57].

Fact-checks play a significant role in the news ecosystem with a shared journalistic purpose of rebuting false claims, therefore they tend to share certain common structured information (called *factors*) in their journalistic practices [57, 60]: a typical fact-check usually introduces the *claim* to be checked and the *claimant* who made the claim, and finally arrives at a *verdict* describing the veracity of the claim. A fact-check could also describe the context where the claim is made, provide evidence to support or attack the claim, etc. An example of a fact-check with three reported factors (claim, claimant and verdict) is shown in Figure 1. This paragraph is excerpted from a fact-check published by PolitiFact in May 2019, titled "D.A.R.E. still thinks marijuana is a dangerous drug for kids", which reached a verdict "False" to the claim "D.A.R.E removed cannabis from its list of gateway drugs." made by "Viral image" on Facebook.

These three factors (claim, claimant and verdict) can summarize the main message of the fact-check for readers on a "too long; didn't read" agenda, and, more importantly, are structured information that can be understood by algorithms for various applications [7, 60, 64, 65], e.g., Google and Bing display these factors as structured snippets for search results that correspond to a fact-check [3, 30].

However, the availability of such factors as structured information is limited. Traditionally, journalists use creative language to embellish their content and attract readers; therefore hiding these factors within the text content of their fact-checks. Structured information has only been recently made available with the global effort on computational journalism [5, 8]. In the context of fact-checking,

a schema named ClaimReview [50] was developed to help annotating these structured information on web pages: A fact-checker can embed the ClaimReview markup in their HTML content of fact-checks, or submit these factors manually through an online ClaimReview markup tool [14]. This is a tedious process for fact-checkers and distracts them from their core job of investigating claims. As a result, Duke Reporters' Lab reported that less than half of their recorded fact-checkers have adopted this schema as of July 2019, and only for a part of their published fact-checks [39]. Therefore, how to extract more structured information from the remaining fact-checks becomes an emerging research concern.

In this paper, we propose the task of *factoring fact-checks* for automatically extracting structured information from fact-checking articles. Leveraging a public dataset of 6K fact-checks available on DataCommons [9], we first conduct an exploratory analysis of the task. We find that reported claimants and verdicts can be mostly found exactly in the text content of fact-checks, while claims are mostly paraphrased from one or more sentences. In addition, we find that these factors are heavily distributed in head and tail sentences of fact-checks, albeit differently between well-known and under-represented fact-checkers. In order to automatically extract these factors, we formulate this task as a sequence tagging problem and conduct several experiments by fine-tuning the state-of-the-art pre-trained BERT models [10]. Our experiments focus on the following research questions:

- How well can models extract claims, claimants and verdicts?
- Can model performance be improved with modifications?
- Can models trained on well-known fact-checkers generalize?

Our experiments demonstrate the performance of BERT models on well-known fact-checkers, especially under the modification made from our empirical observations. Although it is challenging for models to generalize to under-represented fact-checkers whose fact-checks are unseen during the training process, we demonstrate promising initial results by conducting additional experiments. As this task directly faces the misinformation problem and therefore requires extremely high accuracy to be fully automated, we discuss several potential applications with the performance as is, e.g., pre-populating ClaimReview markups in a human-in-the-loop process [14], or supporting other downstream tasks of computational fact-checking [7, 60, 64, 65].

To summarize, we make the following contributions:

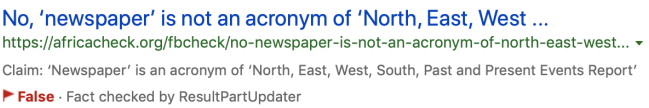
- Defined the task of *factoring fact-checks* to assist fact-checkers.
- Explored existing fact-checks and their reported factors to understand the challenges involved in the task.
- Modified and fine-tuned BERT models and conducted extensive experiments to approach the task.

The rest of the paper is organized as follows: § 2 introduces background and positions our work around related areas, § 3 explores data, § 4 formulates the task and introduces the model, § 5 reports the results of our experiments, § 6 discusses potential applications, limitations and future work, and finally concludes.

2 BACKGROUND

Fact-checks have been around since early 2000s and came to a broader public consciousness in 2016 [16], directly in response to the misinformation epidemic [1, 17]. In this section, we briefly introduce the background of misinformation and fact-checking, and situate our task in the broader scope of computational linguistics.

(a) Google snippet. A fact-check from PolitiFact is displayed.



(b) Bing snippet. A fact-check from Africa Check is displayed.

Figure 2: Structured fact-check snippets. When searching a made-up explanation of the word "newspaper", fact-checking features are displayed by both Google and Bing.

2.1 Misinformation and Fact-Checking

There have been significant efforts on understanding misinformation from both researchers [34] and practitioners [62]. Early work on misinformation discussed its psychological foundations [42, 49, 61], economic incentives [1, 13] and social impact [27, 59]. Meanwhile, studies from the computational community were mostly focused on detecting misinformation on the web [51]: these studies modeled the problem as a classification task and utilized various features (e.g., stylistic differences [12, 46, 58], public responses [27–29]) to access the trustworthiness of information. Although solid results were reported, these models heavily rely on the inflammatory and sensational language used by misinformation to instigate its readers, instead of verifying the information per se [45].

Fact-checking, as a complementary approach (arguably) orthogonal to stylistic features and public opinions, rebuts misinformation by checking the veracity of individual factual claims [16, 57, 60]. Fact-checking is a time-consuming process done by journalists collecting evidence and writing articles (i.e., fact-checks). Although recent studies explored the potential of adding automation to several stages of this process (e.g., discovering check-worthy claims from large corpus [20–22], checking existing knowledge graphs for evidence [7, 64, 65]), high quality fact-checks are still scarce.

As the availability of fact-checks is limited, the utilization of fact-checks becomes of vital importance. Platforms have developed a variety of applications to utilize fact-checks, e.g., downranking [6], moderation [25, 26]. An application promulgated by search engines (i.e., Google [30] and Bing [3]) is enriching result pages with fact-checking features to maximize the exposure of high quality fact-checks by displaying structured snippets of their factors when searching a relevant query [60]. Figure 2 shows examples of such structured snippets from Google (Figure 2a) and Bing (Figure 2b) when searching a made-up explanation of the word "newspaper".

Such application relies on structured factors reported by fact-checkers. However, less than half of fact-checkers reported factors for their fact-checks as of July 2019, according to Duke Reporters' Lab [39]. Our task provides an upstream support for this process,

as we expect more structured factors can be obtained with the help of factoring fact-checks. Specific use cases and other downstream applications that can benefit from our task are discussed in § 6.

2.2 Extracting, Mining and Verifying Claims

There are several existing tasks in the broader scope of computational linguistics that are related to our task.

Claim extraction (or *detection, identification*) is a task of finding factual claims in an article. The target claim could be either context dependent [36] or independent [38]. In the fact-checking context, ClaimBuster is a popular system that ranks claims by "checkworthiness" in news articles or political speeches [21, 22]. In § 5, we apply this tool as a baseline method for the claim factor, and show that the most "checkworthy" claim in a fact-check is often not the fact-checked one.

Argument mining is a more general task of labeling arguments in an article, including both claim extraction and relationship (e.g., supporting or attacking, being premise or evidence) prediction between arguments [4]. This task is often formulated as a sentence classification or boundary detection problem, and its research has been applied to many forms of literature, including persuasive essays [11, 52], scientific publications [54], Wikipedia articles [2, 48], debate scripts [19, 43], etc. Our task can be approximately viewed as a specific case of relationship prediction for fact-checks, as we aim to extract the target claim and the verdict that supports or attacks the claim, except with different task formulation and context. In addition, we focus on only two factors instead of labeling the argument structure of the entire fact-check.

Claim verification is a recently proposed task, that takes a claim and the corresponding context where claim is made as inputs, and outputs a binary verdict on whether the context supports or rejects the claim [55, 56]. In our task, the claim is unknown and the verdict is free text: we take a fact-check as input, and output the claim and its verdict simultaneously, along with its claimant.

3 DATA EXPLORATION

DataCommons hosts a dataset of URLs for fact-checks [9]. We use an internal Google tool to extract main article text from these URLs, filter out non-English fact-checks and keep the remaining 6,216 ones for our data exploration. These fact-checks are usually long articles, with 1,038 words and 24.4 paragraphs on average.

Each fact-check is labeled with its three factors: claim, claimant and verdict, reported by its fact-checker. Claims are usually one or two sentences, with 22.2 words on average. Claimants are mostly names of people or organizations, with 2.2 words on average. Verdicts are mostly adjective phrases and 2.5 words on average.

In this section, we explore several questions of the fact-check dataset to understand our task.

3.1 Who Are the Fact-Checkers?

We first answer the question *who the fact-checkers are.* As shown in Figure 3: the number of fact-checks follows a power law distribution over fact-checkers, where top 19% (5/27) of fact-checkers publish 94% (5,868/6,216) of fact-checks and 40% (11/27) of fact-checkers have reported only a single fact-check. Notably, PolitiFact dominates this dataset with 63% (3,915/6,216) reported fact-checks.

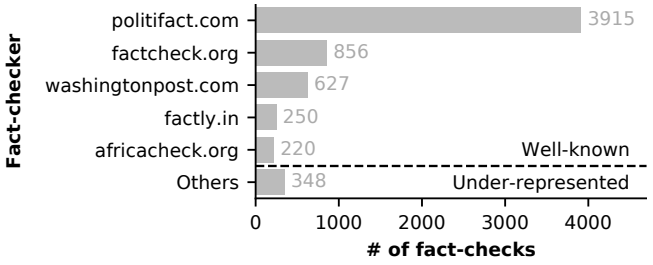


Figure 3: Who are the fact-checkers? the number of fact-checks follows a power law distribution over fact-checkers, where "well-known" (top 5) fact-checkers publish 94% of fact-checks.

In the rest of the paper, we refer to the top five fact-checkers as "well-known" ones, as they are reputed journalistic organizations that are heavily focused on fact-checking and have specialized "fact-check" columns on their websites. Among them, three fact-checkers (PolitiFact, FactCheck.org and the Washington Post) are US-based and have been contributing to fact-checks for more than 10 years. The other two fact-checkers are also well-known in their respective markets: Factly.in is from India and Africa Check is from Africa, both founded in 2012. The rest of the 22 fact-checkers are "under-represented" (i.e., with few samples) in this dataset. These fact-checkers could be **a)** newly established fact-checking teams from reputed journalistic organizations (e.g., Australian Associated Press), **b)** well-known news agencies doing occasional fact-checking (e.g., CNN, the Verge), or **c)** new coming agencies for localized fact-checking (e.g., factcheckNI for Northern Ireland). In § 5, we discuss how this split affects our experimental design.

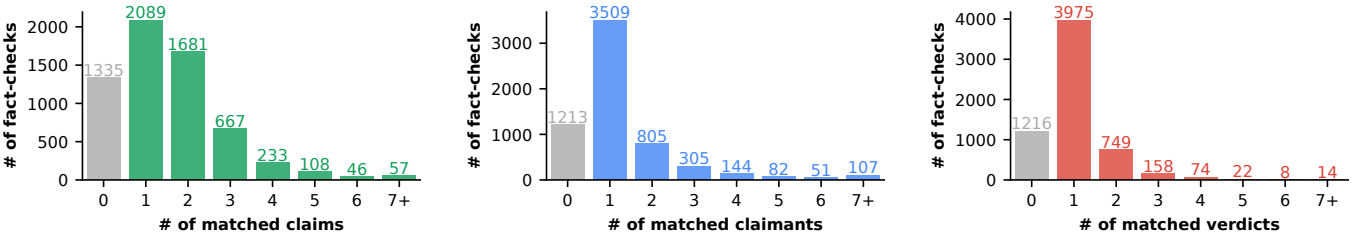
Overall, this dataset records more than one third of all fact-checkers that verified signatories of the International Fact-Checking Network (IFCN) code of principles [24], and contains all the well-known fact-checkers except Snopes, therefore it is a reasonably representative sample of the current fact-checking ecosystem.

3.2 Can Factors Be Found in the Fact-Check?

Next, we answer the question *if factors can be found in the fact-check article.* In the introductory example shown in Figure 1, the claim "D.A.R.E. removed cannabis from its list of gateway drugs." and verdict "False" can be both matched in the text content of the fact-check, while the claimant "Viral image" cannot. Instead, the fact-check uses "Facebook post" as the claimant.

To answer this question for all fact-checks in the dataset, we start with *exact string matching* between factors and fact-checks. We find that most verdicts (76%, 4,743/6,216) and claimants (80%, 4,988/6,216) can be matched in the fact-check, while claims are more difficult and match only 32% (2,000/6,216). This number is counter-intuitive for claims, as we expect the fact-checked claim should appear more frequently in the text content of fact-checks.

After reading through several examples of claims and fact-checks, we find that although most claims are not exactly repeated in the text content of fact-checks, they are mostly paraphrased from one or more sentences, e.g., in Figure 2, the reported claim from PolitiFact "Says the word newspaper stands for 'north, east, west, south, past and present event report.'" is paraphrased to "'newspaper' is an



(a) Can claims be found? 79% claims can be matched in fact-checks and 45% are matched more than once.

(b) Can claimants be found? 80% claimants can be matched in fact-checks and 24% are matched more than once.

(c) Can verdicts be found? 80% verdicts can be matched in fact-checks and 16% are matched more than once.

Figure 4: Can factors be found in the fact-check? Around 80% of factors can be roughly matched in fact-checks based on a fuzzy matching rule, and most of them are matched only once.

acronym for 'North, East, West, South, Past and Present Report.'" in the fact-check. To find these paraphrased factors, we develop a fuzzy matching rule: we first traverse each paragraph and keep the ones that contain at least a certain percentage of the reported factors, and then find the *minimum window substring*[1] [35] of the overlap from the paragraph as the approximate match for the factor.

To choose a reasonable threshold for above-mentioned percentage, manually check 20 fact-checks and matched factors under a spectrum of thresholds. We find that setting the overlap threshold around two thirds gives the best match without introducing false positives (i.e., incorrectly matched factors). We apply the same threshold for all factors: For claims, this allows us to ignore certain paraphrase and conjugation (e.g., "says/said/saying"); For claimants and verdicts, this threshold represents an exact match for short factors less than or equal to two words, e.g., "False", "Mostly false", "John Doe";[2] and allows some flexibility for more than two words, e.g., the claimant "John M Doe" can be matched without the middle name "M". After matching, we again manually check 100 random samples and find no false positives.

Figure 4 shows the histograms of matched factors. Note that factors can be matched zero times or more than once. Under fuzzy matching, more claims (79%, 4,881/6,216) can be matched in fact-checks and 45% (2,792/6,216) are matched more than once (Figure 4a). The number for matched claimants is also slightly increased, with 80% (5,003/6,216) matched and 24% (1,494/6,216) matched more than once (Figure 4b). Matched verdicts are roughly the same, 80% (5,000/6,216) verdicts are matched and 16% (1,025/6,216) are matched more than once (Figure 4c). In general, this observation suggests that around 80% of factors can be roughly matched in fact-checks, the remaining ones are framed to the extend that exceeds our allowed threshold.

3.3 Where Are the Factors in the Fact-Check?

Our final question is *where the factors are in the fact-check* if they are matched. To answer this, we normalize the locations of matched factors by the number of words in the fact-check. This results in a relative position measure ranging from 0 to 1 for each factor in each fact-check, where 0 represents the head and 1 represents the tail of the fact-check. Then, we estimate the probability density functions (PDFs) for the relative position measure of each factor, and plot them in Figure 5. Recall that well-known fact-checkers publish 94% of fact-checks, the PDFs of relative factor positions from all fact-checks would mostly reflect the distribution of well-known ones. Therefore, we estimate separate PDFs for well-known and under-represented fact-checkers in addition to the overall PDFs to compare their difference.

As shown in Figure 5, there are two high-level observations that apply in general for all factors: **a)** factors distribute heavily on head (<0.15) and tail (>0.85) sentences of fact-checks, an observation that we later utilize for model design in § 4, and **b)** overall distributions (thick grey lines) are similar to the distribution of well-known fact-checkers (colored dashed lines) as expected.

For claims, Figure 5a shows that claims are distributed differently for fact-checks from well-known fact-checkers and under-represented ones (Mann-Whitney $U = 2.1 \times 10^{8***}$):[3] Claims are found on both head (<0.15, 41%) and tail (>0.85, 28%) sentences from well-known fact-checkers, but mostly head (<0.15, 53%) sentences from under-represented fact-checkers. This is because well-known fact-checkers usually start their fact-check with an introductory paragraph and end with a concluding one, both of which likely describes the checked claim. Under-represented fact-checkers usually do not write the latter part, and most fact-checks only introduce the claim in the beginning.

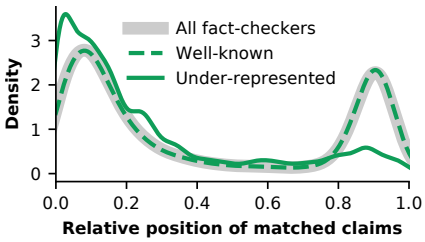
Claimants are distributed similarly between well-known and under-represented fact-checkers. The distribution for well-known ones is slightly shifted to the left ($U = 1.1 \times 10^{7***}$), as shown in Figure 5b. All claimants are heavily distributed on head sentences of the fact-checks (<0.15, 62% for well-known fact-checkers and 66% for under-represented ones).

For verdicts, Figure 5c shows different distributions of relative positions between well-known and under-represented fact-checkers ($U = 5.6 \times 10^{5***}$). For well-known fact-checkers, the verdicts are matched mostly on the very end of the fact-check (>0.85, 73%), usually in the last several words. This reflect the journalistic style of well-known fact-checkers, especially PolitiFact and The Washington Post, whose verdicts are based on their own rating systems and are reached in the very end. These verdicts usually act as complements to linking the verb to be, e.g., "We rate it Pants on Fire!" by

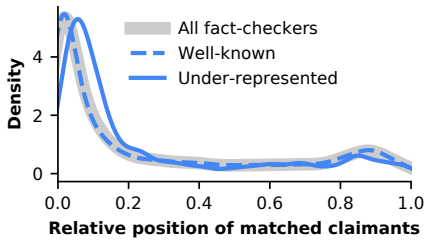
[1] Comparison at word-level and case-insensitive.

[2] 86% (5,338/6,216) of claimants and 79% (4,915/6,216) of verdicts are of less than or equal to two words.

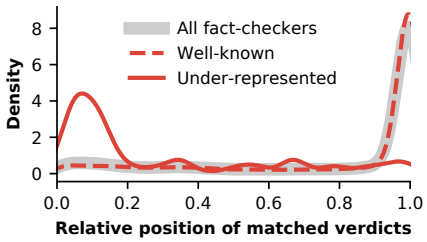
[3] $^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$



(a) Where are the claims? Claims from well-known fact-checkers are distributed on both head (<0.15, 41%) and tail (>0.85, 28%) sentences of fact-checks, but mostly heads (<0.15, 53%) from under-represented fact-checkers. The difference is significant ($U = 2.1 \times 10^{8***}$).



(b) Where are the claimants? Claimants from well-known and under-represented fact-checkers are distributed similarly, heavily on head sentences (<0.15, 62% for well-known ones and 66% for under-represented ones). The difference is significant ($U = 1.1 \times 10^{7***}$).



(c) Where are the verdicts? Verdicts from well-known fact-checkers are distributed on the very end of the fact-check (>0.85, 73%), but mostly head (<0.15, 59%) sentences from under-represented fact-checkers. The difference is significant ($U = 5.6 \times 10^{5***}$).

Figure 5: Where are the factors in the fact-check? Factors distribute heavily on head (<0.15) and tail (>0.85) sentences of fact-checks, and overall distributions are similar to the distribution of well-known fact-checkers.

PolitiFact and "It earns Four Pinocchios." by the Washington Post. On the contrary, under-represented fact-checkers tend to introduce their verdicts in the beginning of the fact-check (<0.15, 59%), and their verdicts are usually adjectives used to modify nouns, e.g., "A false rumor claims...". These fact-checks also have no concluding paragraphs in the end, therefore verdicts can only be found in the head sentences.

4 TASK AND MODELS

Data exploration shows that most factors can be fuzzily matched in fact-checks. In this section, we formulate the task of factoring fact-checks as a computational linguistic task and introduce our models for the task.

4.1 Task Formulation

In general, certain linguistic patterns can be found when a factor appears in the fact-check. These linguistic patterns can be roughly categorized to two types: the languages used by the factor *per se* and its surrounding *context*.

4.1.1 Linguistic patterns. Claims usually appear *following* certain verbs as context, e.g., "(someone) claimed...", "(someone) said...", "(an image/post) shows...", etc., although the beginning of the claim might not be directly adjacent to these verbs (as additional information can be inserted, e.g., "claimed, at somewhere, that..."). In addition, the languages used by claims are mostly factual sentences, and sometimes contains quotation marks, numbers or statistics, entities, etc., e.g., "A 1988 exorcism took place in the only house still standing after Hurricane Ike." is a claim checked by PolitiFact which contains the year "1988" and the entity "Hurricane Ike".

Claimants, opposite to claims, are usually *followed* by above-mentioned verbs as context (i.e., "someone", "an image/post" in previous examples), and they are usually named entities such as persons (e.g., politicians, celebrities) or organizations (e.g., news agencies, social media platforms).

Verdicts are mostly adjective words or phrases that describe veracity, e.g., "true", "mostly true", "false", or phrases from specialized ratings systems, e.g., "pants on fire" by PolitiFact and the "Pinocchio" system by the Washington Post. Occasionally, verdicts can also be descriptive text, e.g., "out of context because...", "outdated as of...". In terms of context, verdicts are often explicitly pointed out by a concluding sentence, e.g., "we rate this claim true", but it could also be embedded in sentences modifying nouns, e.g., "a false rumor claims...".

4.1.2 Sequence tagging. Combining these linguistic patterns together, it is implausible to build and expensive to maintain a rule-based system that matches patterns to extract these factors. Instead, we formulate the extraction task as a computational linguistic task, *the sequence tagging problem*, that can be approached by probabilistic models.

In general, the goal of the sequence tagging problem is to probabilistically estimate the distribution of labels to each word in a sequence. This is a common problem that are shared by a number of existing tasks: Part-of-speech tagging task assigns a part-of-speech label (e.g., noun, verb) to each word in a sentence [40, 47]; Named entity recognition task assigns a entity label (e.g., person, organization, country) to each word in given text [33, 41]; Extractive summarization task assigns a binary label representing if a word should belong to the summary [18, 63], etc.

The expected input of the sequence tagging problem is a sequence of words, and the output is an equal-length sequence of labels (i.e., a label for each word). In our task, there are three positive labels (i.e., claim, claimant and rating) representing if a word should belong to a factor, and a negative label representing a word not belonging to any factors.

4.1.3 Generating ground-truth labels. Recall that fact-checks and its factors are available in our data and we developed a fuzzy matching rule in § 3. This information can be used to generate equal-length sequences as labels for the sequence tagging problem.

For a fact-check, we first initialize an equal-length sequence with all negative labels, and then traverse through all matched factors and replace negative labels with positive ones at selected indexes. We propose two methods of selecting indexes: A *fluent tagger* that selects indexes of the entire span of matched factors and a *concise tagger* that selects indexes of only the overlapping words. As shown in Figure 6, the fact-check is "John M Doe made a false claim that

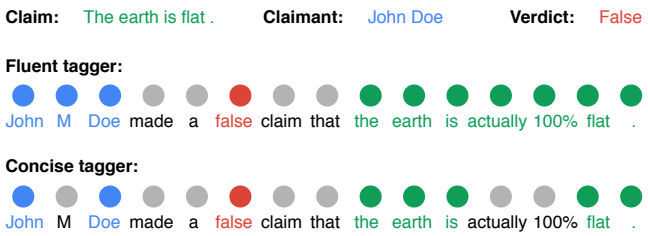


Figure 6: Fluent and concise tagger. The fluent tagger selects indexes of the entire span of matched factors, i.e., "John M Doe" as the claimant and all words starting from "the" as the claim; The concise tagger indexes of only the overlapping words, i.e., skipping non-overlapping words "M" for the claimant and "actually 100%" for the claim.

the earth is actually 100% flat." and its three factors are "The earth is flat" as the claim, "John Doe" as the claimant and "False" as the verdict.[4] The fluent tagger labels "John M Doe" as the claimant and all words starting from "the" as the claim, while the concise tagger skips non-overlapping words "M" for the claimant and "actually 100%" for the claim. Intuitively, the fluent tagger is focused more on readability as it generates continues phrases as ground-truth, but it also inevitably includes unessential details in the sequence. On the contrary, the concise tagger is focused more on brevity as it only selects essential words of factors, but the results could be less readable if the matched factors miss several words.

4.2 Models

Sequence tagging is a traditional computational linguistic problem that has been studied for decades from early statistical methods (e.g., Hidden Markov models [31], conditional random fields [32]) to current neural architectures (e.g., recurrent neural network [15]). To date, the most applicable models usually leverage transfer learning following the pre-training/fine-tuning paradigm. In short, during pre-training process, models are trained on unlabeled data with certain objectives, and during fine-tuning, models are initialized with pre-trained parameters and then re-train on labeled data over specific tasks.

4.2.1 BERT models. BERT (Bidirectional Encoder Representations from Transformers) is a recently developed model [10]. Its pre-training objective is to predict missing (i.e., masked) words in a sentence conditional on both left and right context, as well as to predict the relationship between sentences (i.e., if a sentence is a next sentence of another). During fine-tuning, it has shown the ability to achieve state-of-the-art performance on many computational linguistic tasks by simply replacing input and output layers.

As BERT provides an easy access to the state-of-the-art without any specific neural architecture design, we experiment with BERT to explore the feasibility of our task. The specific framework of model is shown in Figure 7. First, our framework feeds the fact-check and its factors to a rule-based tagging pipeline described in § 4.1.3 to generate sequences and labels for the BERT model; Then, our framework passes these sequences and labels to the BERT

[4]This one sentence fact-check is hypothetical to demonstrate how indexes are selected. In general, it is unlikely that all factors can be found in a single sentence.

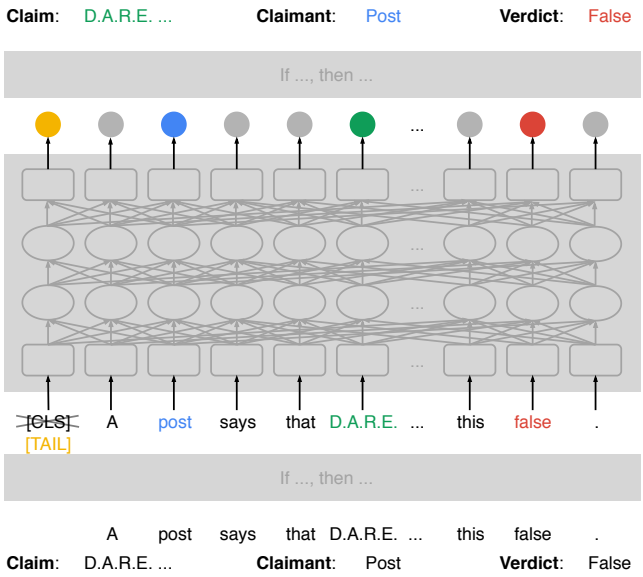


Figure 7: The framework for factoring fact-checks. First, the framework feeds the fact-check and its factors to a tagging pipeline; Then, the framework passes sequences and labels to the BERT model and uses cross entropy loss to fine-tune the model; Finally, the predicted labels is fed to a recovery pipeline to obtain factors. We feed our inputs one paragraph at a time and chunk paragraphs exceeding maximum length. lead token [CLS] is replaced with paragraph positions [HEAD], [BODY] or [TAIL].

model, obtains the activations of its last layer, and feeds them to an output layer for predictions. During the process, cross entropy loss is used to propagate errors and fine-tune the model [44]; Finally, the predicted labels are fed to a rule-based recovery pipeline to concatenate words and predict factors.

Note that BERT is designed for short sequences with a default maximum sequence length of 512 while fact-checks are in general longer sequences.[5] A common strategy dealing with long sequences is to truncate after the maximum length, because the head of the sequence usually captures its essence for a number of tasks, e.g., summarization, classification. However, truncation is not fit for our task as factors can be matched anywhere in the text content of a fact-check, as shown in Figure 5. Therefore, we run our framework on paragraph level and feed our inputs one paragraph at a time. If a paragraph alone is longer than the maximum length, we chunk the paragraph to sub-paragraphs and feed them to BERT in order.

4.2.2 Modification on lead token. In the original BERT, the lead token of the input sequence is a special token [CLS], whose final hidden state is used as the aggregated sequence representation for the classification task. This design choice does not encode any external information of the input sequence to the model, instead, it learns a representation for [CLS] using the information from the words in the sequence *per se*.

[5]1,038 words on average, and the length would increase with the wordpiece tokenizer from BERT.

Table 1: Test on well-known fact-checkers. Fact-checks from this test set are published by the same set of fact-checkers in the train set. BERT models significantly outperforms baseline methods, and replacing [CLS] with paragraph positions help to improve the overall performance of models.

Lead token	Tagger	**Claim** ROUGE-1			**Claimant** ROUGE-1			**Verdict** ROUGE-1		
		F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
Baseline		.183 (.183)	.300 (.300)	.141 (.141)	.237 (.237)	.181 (.181)	.352 (.352)	.660 (.660)	.638 (.638)	.702 (.704)
[CLS]	Fluent	.636 (.853)	.669 (.897)	.633 (.850)	.769 (.894)	.803 (.934)	.759 (.883)	.931 (.975)	.934 (.979)	.930 (.974)
	Concise	.592 (.864)	.615 (.897)	.596 (.870)	.784 (.907)	.789 (.913)	.783 (.906)	.938 (.971)	.940 (.973)	.938 (.970)
Paragraph position	Fluent	**.638** (.854)	**.674** (.902)	**.637** (.853)	**.794** (.889)	**.821** (.919)	**.789** (.884)	**.940** (.978)	**.942** (.980)	**.939** (.978)
	Concise	**.646** (.866)	**.664** (.889)	**.652** (.873)	**.839** (.928)	**.852** (.943)	**.834** (.923)	**.941** (.975)	**.944** (.979)	**.940** (.974)

As we feed fact-checks to the our framework paragraph by paragraph, applying the original BERT with [CLS] lead token would loose the external information of the relative positions of paragraphs. This information, however, is highly associated with the appearance of factors: As shown in Figure 5, head and tail sentences of a fact-check are much more likely to contain factors.

To utilize this external information, we make a simple modification of BERT by replacing the uniform [CLS] token with a *paragraph position* token, which could be either [HEAD], [BODY] or [TAIL] to represent the relative position of a paragraph. We also associate these tokens with three corresponding labels in the output sequence (as shown in Figure 7). We expect this modification can help the BERT model to learn better representations for each paragraph based on its location, and therefore improving the ability to tag factors in the paragraph.

5 EXPERIMENTS

Under the BERT framework described in Figure 7, we conduct several experiments to explore the feasibility of factoring fact-checks. Results reported in this section are based on the same public dataset in section 3 to ensure the reproducibility of our results.

5.1 Setup

Our experimental setup is focused on answering the following questions: *How well can models extract claims, claimants and verdicts respectively? Can model performance be improved with modifications made from our empirical observations? Can models trained on well-known fact-checkers generalize to under-represented ones?*

5.1.1 Data splitting. A common data splitting strategy is randomly sampling portions of the entire dataset to train, dev and test sets. However, in § 3, we show that **a)** fact-checks follow a power law distribution over fact-checkers, where the well-known ones published most part of fact-checks (Figure 3), and **b)** well-known and under-represented fact-checkers have different journalistic styles, measured by the relative positions of factors in fact-checks (Figure 5). This suggests that models learned and evaluated by using the common data splitting strategy would mostly reflect the performance on well-known ones.

To comprehensively understand the performance of models on both well-known and under-represented fact-checkers. We split our data in the following way: First, we keep fact-checks from well-known fact-checkers and do a 80%/10%/10% (4,694/587/587 samples) random split to train, dev and test sets (i.e., the the same way as the common strategy), where the train set is used to fine-tune model parameters, the dev set is used for early stopping to prevent overfitting, and the test set is used to evaluate the performance of models on well-known fact-checkers; Second, we use the remaining fact-checks from under-represented fact-checkers (348 samples) as a second test set. This test set is used for evaluation only and invisible during the training process, which allows us to understand how models can generalize across different journalistic styles.

5.1.2 Baseline methods. Intuitive baseline methods are also experimented to demonstrate the non-triviality of the problem.

For claims, our baseline methods leverage a popular system named ClaimBuster, which scores the "checkworthiness" of sentences in an article. The hypothesis is that the fact-checked claim might also has the highest score of checkworthiness in a fact-check. Therefore, we label the entire sentence with the highest score as the claim.

For claimants, our intuition contains two points: **a)** claimants are most likely to be a person or organization, and **b)** they are also likely to be heavily mentioned in the fact-check. Therefore, we run our fact-checks through a named entity recognition pipeline implemented in spaCy [23], and label the most frequent person or organization as the claimant.

For verdicts, we take a simplistic approach. For each fact-check, we find the most frequently mentioned of all verdicts from the train set and use it as the predicted verdict.

5.1.3 Model variants. Besides the baseline methods, we compare four variants of models: **a)** the original BERT ([CLS] as the lead token) and our modified version (paragraph positions as the lead token) to understand whether replacing lead tokens improves model performance, and **b)** input sequences tagged by fluent and concise taggers.

5.1.4 Hyperparameters and fine-tuning process. All reported results use the same neural architecture, hyperparameters and vocabularies (except for lead tokens) as the uncased base BERT model.[6] For lead tokens, we label the first 3 paragraphs in a fact-check as [HEAD], the last 3 paragraphs as [TAIL] and remaining as [BODY].[7]

The dev set is used for early stopping, and we observe that models achieve lowest loss around 10K steps in general. For each

[6] Cased and large BERT models are also experimented but yields similar results, therefore we omit the results in this paper.

[7] Fact-checks have 24 paragraphs on average, and we split them on the 0.15/0.85 threshold shown in Figure 5.

Table 2: Test on under-represented fact-checkers. Fact-checks from this test set are published by different fact-checkers than the ones in the train set. The performance heavily deteriorates comparing to well-known fact-checkers. The paragraph tokens still helps to improve both precision and recall for most cases.

Lead token	Tagger	**Claim** ROUGE-1			**Claimant** ROUGE-1			**Verdict** ROUGE-1		
		F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
Baseline		.175 (.175)	.372 (.372)	.122 (.122)	.132 (.132)	.114 (.114)	.204 (.204)	.392 (.392)	.385 (.385)	.409 (.409)
[CLS]	Fluent	.444 (.725)	.483 (.788)	.443 (.724)	.264 (.567)	.364 (.782)	.236 (.506)	**.429** (.806)	**.484** (.910)	**.421** (.792)
	Concise	.386 (.713)	.406 (.748)	.406 (.749)	.323 (.650)	.379 (.764)	.304 (.612)	.451 (.832)	.484 (.892)	.446 (.821)
Paragraph position	Fluent	**.519** (.728)	**.566** (.794)	**.517** (.725)	**.377** (.635)	**.510** (.859)	**.342** (.576)	.367 (.733)	.451 (.902)	.359 (.718)
	Concise	**.527** (.738)	**.532** (.744)	**.559** (.781)	**.462** (.709)	**.549** (.843)	**.436** (.670)	**.473** (.832)	**.520** (.914)	**.467** (.822)

experiment, the fine-tuning process runs within an hour on a single TPU node.

5.1.5 Evaluation metrics. The focus of the evaluation is *how well our models tag claims, claimants and verdicts respectively*. Therefore, we evaluate the performance of each factor separately by using ROUGE scores [37]. Briefly, ROUGE scores are extensively used evaluation metrics that compare N-gram overlaps between references and predictions, e.g., ROUGE-1 refers to unigram, ROUGE-L refers the longest N-gram. References in our experiments are recovered labels generated in subsubsection 4.1.3, and ROUGE scores are undefined (i.e., uncounted) when references are empty. Each ROUGE score measure contains F1, precision and recall scores, where precision measures how much of the overlap is captured by the prediction, recall measures how much of the overlap is captured by the reference, and F1 is the harmonic mean of the precision and recall. In the rest of the paper, we report F1, precision and recall scores in ROUGE-1.

Note that if a factor is not tagged by the model, i.e., the prediction is empty, the default ROUGE score of the factor is 0, which is a *tight score* that heavily lowers the average. Depending on the application (as we will discuss in § 6), a *loose score* can also be useful that only evaluates when a factor is tagged, i.e., the prediction is non-empty. The loose score optimistically measure how correctly our models tag a factor as long as the factor is tagged, and the tight score is the product of the loose score and the tagged percentage. In the remaining tables, we report results in a format of: tight score (loose score) in a cell.

5.2 Results

We first explore the overall feasibility of our task, and then discuss each of above-mentioned questions. Due to space limit, tables in this section are excerpted to highlight our observations.[8]

5.2.1 Overall performance. Table 1 reports results on the test set of well-known fact-checkers. Fact-checks from this test set are published by the same set of fact-checkers in the train set.

As shown, there is a large gap between all BERT models and baseline methods. Note that baseline methods always return non-empty predictions, therefore the tight score is strictly equal to the loose score.

[8]Full comparison table of all experiments with ROUGE-1/ROUGE-L tight/loose F1/precision/recall scores can be found in Supplementary Materials.

For claims, BERT models achieve tight scores of 0.59-0.65 and loose scores of 0.85-0.87, with 69%-75% of claims tagged overall. Tagging claimant is a relatively easy task, as BERT models achieve tight scores of 0.77-0.84 and loose scores of 0.89-0.93, with 86-90% claimants tagged. BERT models archive highest scores on tagging verdicts, where the tight scores are 0.93-0.94 and loose scores are 0.97-0.98, with 96%-97% of verdicts tagged.

This demonstrates the overall feasibility of factoring fact-checks, at least for well-known fact-checkers.

5.2.2 Paragraph position as lead token. Table 1 suggests that replacing [CLS] with paragraph positions helps to improve the overall performance of models. Comparing tight scores of models with the same tagger but different lead tokens, the ones with paragraph position as lead tokens outperform in both ROUGE-1 precision and recall (and therefore F1) than the ones with [CLS]. This improvement is consistent across all three factors and both two taggers, although sometimes by a small margin (e.g., F1 for verdict with concise tagger improves from 0.938 to 0.941 by 0.003), sometimes by a larger one (e.g., F1 for claim with concise tagger improves from 0.592 to 0.646 by 0.054).

5.2.3 Generalization. Table 2 reports scores of evaluation metrics on the test set of under-represented fact-checkers. Fact-checks from this test set are published by different fact-checkers than the ones in the train set, therefore their journalistic styles are completely unseen in the training process.

As shown, the performance heavily deteriorates comparing to the results from well-known fact-checkers. For claims, BERT models achieve tight scores of 0.39-0.44, 54%-61% tagged percentages with [CLS], and 0.52-0.53, 71%-72% tagged percentages with paragraph positions. The performance of tagging claimant drops even more, where the best model achieves a tight score of 0.46 and a loose score of 0.71, with 65% claimants tagged. Similarly, the best model of tagging verdict achieves a tight score of 0.47 and a loose score of 0.83, with 57% of verdicts tagged.

Note that using paragraph positions as lead tokens still helps to improve both precision and recall for tagging claims and claimants, and verdicts with the concise tagger, with verdicts with fluent tagger being the only exception.

5.2.4 Improving generalization. In order to improve the performance on under-represented fact-checkers. We conduct an additional experiment by mixing half of the test set (174 samples) to the train set and retrain the model, which makes the pattern of this test

Table 3: Test on under-represented fact-checkers by mixing half to train set. Model performance of tagging claims is not improved, the performance of tagging claimants and verdicts is improved. This demonstrates a promising direction: as more and more fact-checkers adopt this scheme, the performance can keep improving when patterns from more fact-checkers are captured by our models.

Train set	Tagger	**Claim** ROUGE-1			**Claimant** ROUGE-1			**Verdict** ROUGE-1		
		F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
Well-known	Fluent	**.519** (.728)	**.566** (.794)	**.517** (.725)	.377 (.635)	.510 (.859)	.342 (.576)	.367 (.733)	.451 (.902)	.359 (.718)
ones only	Concise	**.527** (.738)	.532 (.744)	**.559** (.781)	.462 (.709)	.549 (.843)	.436 (.670)	.473 (.832)	.520 (.914)	**.467** (.822)
under-repre	Fluent	.495 (.761)	.540 (.830)	.489 (.752)	**.550** (.717)	**.639** (.832)	**.528** (.688)	**.475** (.712)	**.573** (.859)	**.469** (.704)
sented mixed	Concise	.519 (.782)	**.544** (.819)	.536 (.807)	**.575** (.781)	**.599** (.813)	**.581** (.789)	**.482** (.797)	**.562** (.931)	.464 (.768)

set partially visible in the training process. As this is a small portion compared to the original train set, the performance on the first test set is largely unchanged. Therefore, we focus on the performance on the remaining fact-check from under-represented fact-checkers.

As shown in Table 3. By mixing half into training, although model performance of tagging claims is not improved, the performance of tagging claimants and verdicts is improved. Also note that only 174 samples are left in this test set, therefore the results are relatively unstable.

This experiment suggests that learning from some new fact-checkers helps to improve the overall performance, at least for claimants and verdicts, even when the sample size is very limited. This demonstrates a promising direction: as more and more fact-checkers adopt this scheme, the performance can keep improving when patterns from more fact-checkers are captured by our models.

5.2.5 Error analysis. There are several common types of error made by our models.

One type of error is not tagging, which results in a tight score of 0 and is uncounted for loose scores. As tagging models estimate likelihood of label distribution at word level, our model can assign all negative labels to all words in the input sequence, therefore resulting empty prediction after the recovery pipeline, especially when the linguistic patterns or context for a factor is unseen or hard to capture. This happens frequently on the test set of under-represented fact-checkers where most of their journalistic styles are different from the known patterns.

The (arguably) worst type of error is tagging the completely wrong factor, which results in near 0 for both tight and loose scores. This is likely to happen when the false positive uses the same words as high-frequency true positives, e.g., the word "false" has a high likelihood of being a verdict. In addition, similar context could also confuse our models to yield incorrect predictions, e.g., words after "[someone] claimed that..." until a punctuation is likely to be assigned with claim labels, even when they are indeed not the fact-checked claim.

Another type of error is under- or over- tagging, which could result in a score from near 0 to near 1. This usually happens when our models miss or extend tags to several words such as "said", "claims", etc. Under- (over-) tagging can also happen when the ground-truth factor is extremely long (short), e.g., when the reported claimant has a long title such as "the 45th and current president of the United States Donald Trump", our models tend to tag only the name of the person "Donald Trump", which results in a low F1 score of 0.33.

Mostly, our models tag perfectly (a score of 1) if the factor follows the majority patterns, e.g., when a claim is quoted by quotation marks, or verdict being distinctive words, e.g., "Pants on Fire", the "Pinocchio" system used by the Washington Post.

6 DISCUSSION

Our experiments demonstrate the overall feasibility of factoring fact-checks, especially for well-known fact-checkers. However, as this task directly faces the misinformation problem, extremely high accuracy is expected to fully automate this process.

Instead, in this section, we discuss several human-in-the-loop applications that can benefit from our models with the performance as is. We also present limitations of our work and potential directions for future research.

6.1 Applications

There are two types of applications that can benefit from our task: Human-in-the-loop user-facing applications, most notably, obtaining ClaimReview markups from fact-checks; And downstream engineer-facing applications, e.g., knowledge-graph based fact-checking, fact-check identification, etc.

6.1.1 Pre-populating ClaimReview. Currently, there are two approaches of providing ClaimReview markups for fact-checkers to Google and Bing.

The first approach is to embed ClaimReview markups in the HTML content of the website. As shown in Figure 8, the fact-checker includes a JSON encoded snippet within the <script> tags of the HTML content, and then search engines can identify this field and store structured information in their database. This process is currently entirely manual, and our models provide an opportunity to optimize it by pre-populating the claim ("claimReviewed"), claimant ("itemReviewed" → "author" → "name") and verdict ("reviewRating" → "alternateName") fields.[9] After pre-population, the fact-checker can then verify and modify the snippet if needed to make sure this information is accurate.

The second approach is to submit required information through a fact-check markup tool developed by Google [14]. As shown in Figure 9, current user interface allow fact-checkers to first submit an URL, and then manually type in the ClaimReview markup information (the claim and verdict are required, the claimant and others are optional) and finally submit. Our task provides a similar opportunity for pre-population by deploying models on the server

[9]Other fields in this markup are either fixed (e.g., "@context", "@type"), trivial to pre-populate (e.g., "datePublished", "url"), or optional.

```
<head>
<title>The world is flat</title>
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "ClaimReview",
  "datePublished": "2016-06-22",
  "url": "http://example.com/news/science/worldisflat.html",
  "claimReviewed": "The world is flat",
  "itemReviewed": {
    "@type": "Claim",
    "author": {
      "@type": "Organization",
      "name": "Square World Society"
    }
  },
  "reviewRating": {
    "@type": "Rating",
    "alternateName": "False"
  }
}
</script>
</head>
```

Figure 8: HTML snippet of ClaimReview markups. The fact-checker adds a JSON encoded snippet within the <script> tags and search engines can identify this field. Our models can be used to pre-populate claim, claimant and verdict fields.

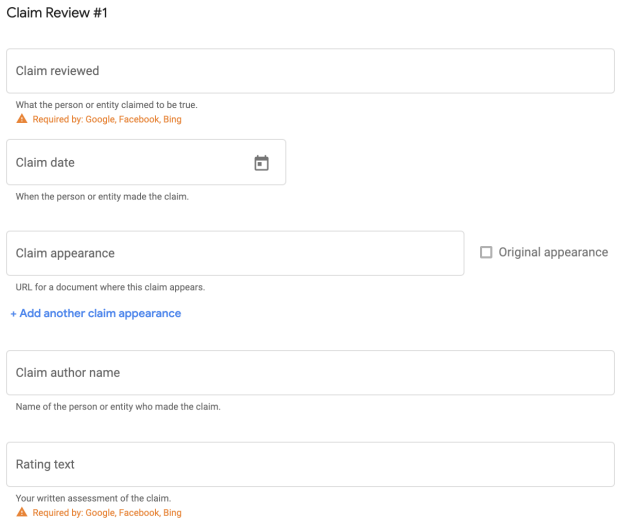


Figure 9: User interface of the fact-check markup tool. Current user interface allow fact-checkers to submit an URL, manually type required information and then submit. Our models provide an opportunity for pre-population.

side: After a URL is submitted, our models can read through the text content of the fact-check and returns tagged factors, and then the fact-checker can verify, modify (if needed) and submit them.

6.1.2 Plug-ins for downstream applications. Our task can provide upstream support for a number of downstream applications on fact-checking, e.g., structured fact-check data for automated fact-checking tasks [7, 64, 65], claims for relevant document retrieval task [60], signals for identifying fact-checks, etc. These applications can directly take the predicted factors from our trained models as input by running models on an archive of existing fact-checks. Alternatively, neural architectures can use our models as a plug-in component and propagate errors for specific tasks. However, the quality of service guarantee is then left for specific downstream applications to consider.

6.2 Limitations and Future Work

Our task focuses on three essential factors in fact-checks: claims, claimants and verdicts. However, fact-checks can contain more factors in their article structures, e.g., (from an argument mining perspective) premise, evidence, etc [4]. Extracting these additional factors, or moreover, parsing the entire argument structure of fact-checks, is a future direction that can benefit not only fact-checking applications, but also document understanding in general.

In addition, we experiment with BERT as a rule-of-thumb tool. Although BERT has shown to achieve state-of-the-art performance on general computational linguistic tasks [10], there is still design space for specific architectures for our task. It is also debatable whether formulating the task as a sequence tagging problem is the optimal solution. Our proposed framework has concatenated pipeline structure (i.e., rule-based tagging and recovering) which can accumulates errors, e.g., as shown in Figure 4, some factors still cannot be found as a result of heavy paraphrasing. Although we also explored the possibility of formulating the task as a language generation problem under the end-to-end encoder-decoder framework [53], we were unable to achieve satisfactory performance, possibly due to limited data, therefore the results are omitted in this paper.

With more data of fact-checks made available, future work can investigate the possibilities of different task formulations, neural architectures, and more automated applications.

6.3 Conclusion

In this paper, we proposed a task of *factoring fact-checks* for automatically extracting structured information (claims, claimants and verdicts) from fact-checks. Our data exploration showed that: **a)** Under fuzzy matching rules, most of factors can be matched from the text content of fact-checks; **b)** These factors are heavily distributed in head and tail sentences of fact-checks, albeit differently between well-known and under-represented fact-checkers. We then conducted several experiments on the feasibility of this task, and showed that: **a)** Overall, BERT models achieve significantly better performance compare to intuitive baseline methods. **b)** Replacing BERT's [CLS] tokens with paragraph positions helps to improve the model performance. **c)** Although it is challenging for models to generalize to under-represented fact-checkers, promising initial results are achieved by mixing half of the fact-checks from under-represented fact-checkers to the training process. Finally, we discussed several potential applications such as pre-populating ClaimReview markups in a human-in-the-loop process and supporting other downstream tasks of computational fact-checking.

We hope that our work plays a role in the progressive process of leveraging fact-checks to improve online information quality: pre-population tools with the help of our models can encourage more fact-checkers to provide ClaimReviews markups, which, in turn, serves as training data to improve the model performance.

REFERENCES

[1] Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives* 31, 2 (2017), 211–36.

[2] Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 251–261.

[3] Bing Blogs. 2017. Bing adds Fact Check label in SERP to support the ClaimReview markup. https://blogs.bing.com/Webmaster-Blog/September-2017/Bing-adds-Fact-Check-label-in-SERP-to-support-the-ClaimReview-markup

[4] Elena Cabrio and Serena Villata. 2018. Five Years of Argument Mining: a Data-driven Analysis.. In *IJCAI*. 5427–5433.

[5] David Caswell and Konstantin Dörr. 2018. Automated Journalism 2.0: Event-driven narratives: From simple descriptions to real stories. *Journalism practice* 12, 4 (2018), 477–496.

[6] Facebook Help Center. 2019. How is Facebook addressing false news through third-party fact-checkers? https://www.facebook.com/help/1952307158131536

[7] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one* 10, 6 (2015), e0128193.

[8] Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational journalism: A call to arms to database researchers. (2011).

[9] DataCommons. 2019. Fact-Check Dataset. https://datacommons.org/factcheck

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[11] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 11–22.

[12] Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 171–175.

[13] Matthew Gentzkow, Jesse M Shapiro, and Daniel F Stone. 2015. Media bias in the marketplace: Theory. In *Handbook of media economics*. Vol. 1. Elsevier, 623–645.

[14] Google. 2019. Fact-Check Markup Tool. https://toolbox.google.com/factcheck/markuptool

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 369–376.

[16] Lucas Graves. 2016. *Deciding what's true: The rise of political fact-checking in American journalism*. Columbia University Press.

[17] Andrew Guess, Brendan Nyhan, and Jason Reifler. 2018. Selective Exposure to Misinformation: Evidence from the consumption of fake news during the 2016 US presidential campaign. *European Research Council* (2018).

[18] Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence* 2, 3 (2010), 258–268.

[19] Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics* 43, 1 (2017), 125–179.

[20] Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. In *Proceedings of the 2015 Computation + Journalism Symposium*.

[21] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. 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*. ACM, 1803–1812.

[22] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, et al. 2017. ClaimBuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1945–1948.

[23] Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* 7 (2017).

[24] IFCN. 2019. Verified signatories of the IFCN code of principles. https://ifcncodeofprinciples.poynter.org/signatories

[25] Shan Jiang, Ronald E. Robertson, and Christo Wilson. 2019. Bias Misperceived: The Role of Partisanship and Misinformation in YouTube Comment Moderation. In *Proceedings of the 13th International AAAI Conference on Web and Social Media (ICWSM 2019)*.

[26] Shan Jiang, Ronald E. Robertson, and Christo Wilson. 2020. Reasoning about Political Bias in Content Moderation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*.

[27] Shan Jiang and Christo Wilson. 2018. Linguistic Signals under Misinformation and Fact-Checking: Evidence from User Comments on Social Media. *Proceedings of the ACM: Human-Computer Interaction (PACMHCI)* 2, CSCW (November 2018).

[28] Zhiwei Jin, Juan Cao, Yu-Gang Jiang, and Yongdong Zhang. 2014. News credibility evaluation on microblog with a hierarchical propagation model. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 230–239.

[29] Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News Verification by Exploiting Conflicting Social Viewpoints in Microblogs.. In *AAAI*. 2972–2978.

[30] Justin Kosslyn and Cong Yu. 2017. Fact Check now available in Google Search and News around the world. https://www.blog.google/products/search/fact-check-now-available-google-search-and-news-around-world

[31] Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language* 6, 3 (1992), 225–242.

[32] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[33] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 260–270.

[34] David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science* 359, 6380 (2018), 1094–1096.

[35] LeetCode. 2014. Minimum Window Substring. https://leetcode.com/problems/minimum-window-substring

[36] Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 1489–1500.

[37] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[38] Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[39] Joel Luther. 2019. Reporters' Lab Launches Global Effort to Expand the Use of ClaimReview. https://reporterslab.org/lab-launches-global-effort-to-expand-claimreview

[40] Lluís Màrquez and Horacio Rodríguez. 1998. Part-of-speech tagging using decision trees. In *European Conference on Machine Learning*. Springer, 25–36.

[41] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.

[42] Raymond S Nickerson. 1998. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology* 2, 2 (1998), 175.

[43] Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument Mining with Structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 985–995.

[44] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is Multilingual BERT? *arXiv preprint arXiv:1906.01502* (2019).

[45] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 231–240.

[46] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2931–2937.

[47] Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*.

[48] Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 440–450.

[49] Robert J Robinson, Dacher Keltner, Andrew Ward, and Lee Ross. 1995. Actual versus assumed differences in construal: "Naive realism" in intergroup perception and conflict. *Journal of Personality and Social Psychology* 68, 3 (1995), 404.

[50] schema.org. 2019. ClaimReview schema. https://schema.org/ClaimReview

[51] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter* 19, 1 (2017), 22–36.

[52] Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics* 43, 3 (2017), 619–659.

[53] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[54] Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards discipline-independent argumentative zoning: evidence from chemistry and computational linguistics. In *Proceedings of the 2009 Conference on Empirical*

Methods in Natural Language Processing: Volume 3-Volume 3. Association for Computational Linguistics, 1493–1502.

[55] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 809–819.

[56] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The Fact Extraction and VERification (FEVER) Shared Task. *EMNLP 2018* 80, 29,775 (2018), 1.

[57] Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. 18–22.

[58] Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 647–653.

[59] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.

[60] Xuezhi Wang, Cong Yu, Simon Baumgartner, and Flip Korn. 2018. Relevant document discovery for fact-checking articles. In *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 525–533.

[61] Andrew Ward, L Ross, E Reed, E Turiel, and T Brown. 1997. Naive realism in everyday life: Implications for social conflict and misunderstanding. *Values and knowledge* (1997), 103–135.

[62] Claire Wardle. 2017. Fake news. It's complicated. *First Draft News* 16 (2017).

[63] Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 985–992.

[64] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward computational fact-checking. *Proceedings of the VLDB Endowment* 7, 7 (2014), 589–600.

[65] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2017. Computational fact checking through query perturbations. *ACM Transactions on Database Systems (TODS)* 42, 1 (2017), 4.