An End-to-End Computational System for Monitoring and Verifying Factual Claims

Kevin Meng
Massachusetts Institute of Technology
Cambridge, MA 02139
mengk@csail.mit.edu

Abstract

This paper introduces `VeriClaim`, a computational fact-checking system that enables the automatic monitoring and verification of factual claims. `VeriClaim` is comprised of two computational modules: the *claim-spotter*, which first identifies check-worthy factual statements from swaths of online information, and the *claim-checker*, which verifies each statement using evidence retrieved from a multitude of knowledge resources. These two components are integrated and deployed in several ways, including a content-rich website and Chrome browser extension. [1]

1 Introduction

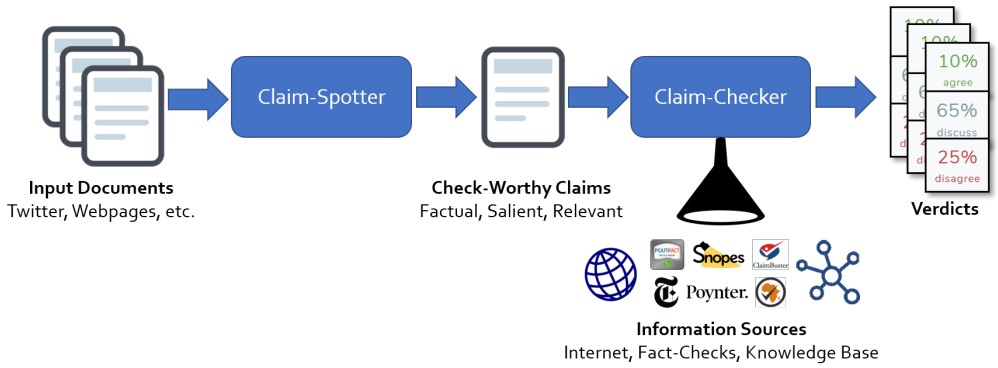


Figure 1: `VeriClaim` System Architecture

Each year, misinformation costs the world economy $80 billion, damages the trust of over 200 million Americans in their government, and improperly sways public opinion on critical topics including political elections, legislative policy, and healthcare. Unfortunately, due to the sheer amount of information produced on modern Internet platforms, as well as limited staffing at fact-checking organizations, many problematic claims go unnoticed and unchecked [1]. Our efforts to mitigate misinformation would benefit greatly from a system that could efficiently identify, track, and check factual statements.

However, this fact-checking manpower shortage is only part of the problem. We also fail to effectively *communicate* fact-checks to the general public. Imagine someone scrolling through their favorite news outlet over morning coffee; they must not only (1) actively question *everything* they read while

[1] Code and additional resources are available at `https://vericlaim.github.io/`

35th Conference on Neural Information Processing Systems (NeurIPS) Workshop on AI for Credible Elections

on the lookout for questionable statements, but also (2) have the dexterity to research and adjudicate the veracity of these claims.

To help combat these issues, we introduce `VeriClaim`: a state-of-the-art automated fact-checker with two key components: claim-spotting and claim-checking. The *claim-spotting* module is tasked with sifting through bodies of text and extracting statements that are both factual in nature and salient to the general public. Given that there is far too much information on the Internet for humans to review manually, effective automation at this step is key. This is followed by *claim-checking*, which verifies claims by cross-referencing repositories of known fact-checks and querying knowledge bases [2]. To the best of our knowledge, there is currently no system able to autonomously monitor large bodies of text and fact-check the salient claims from end-to-end.

In the long-run, it is our hope that `VeriClaim` can benefit both fact-checkers and the general public. On one hand, `VeriClaim` would help fact-checkers focus on what they do best (performing novel, in-the-field research) by automating grunt work. For the general public, we hope that the various deployed applications will help bring fact-checks *to the users* and thus minimize the distance between them and the truth.

2 Technical System Overview

As Figure 1 suggests, our pipeline is divided into two components: claim-spotting and claim-checking. The input consists of any type of textual document such as tweets, news articles, webpages, etc., and the output is a series of verdicts that describe the estimated factuality of each check-worthy claim in the input.

2.1 Claim-Spotting Framework

The claim-spotting framework is implemented exactly as detailed in *Gradient-Based Adversarial Training on Transformer Networks for Detecting Check-Worthy Factual Claims* [3]. They provide a convenient publicly-accessible API endpoint [2] for queries.

To summarize briefly, the authors formulate claim-spotting as a classification task; that is, a statement can be categorized as either check-worthy or non-check-worthy. They find that integrating a novel gradient-based adversarial training algorithm on top of such a model yields better results due to regularization. Their work achieves state-of-the-art performance on a challenging *politics-based* benchmark dataset and is thus a good fit for this work.

2.2 Claim-Checking Framework

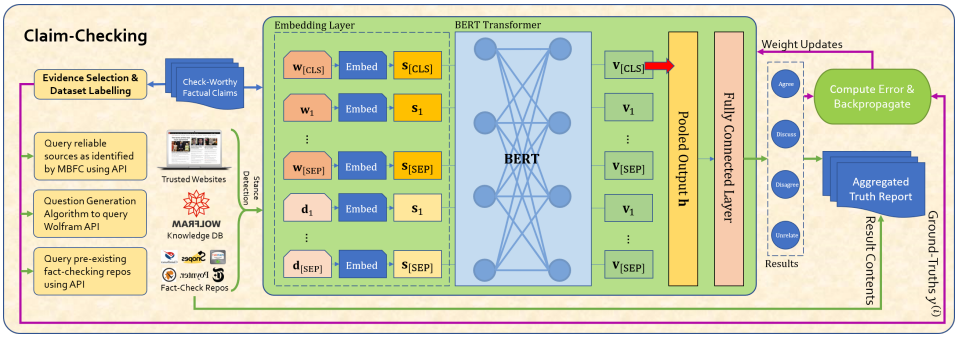


Figure 2: Claim-Checking Framework

We frame claim-checking as process where a system scours the Internet for documents that are related to a given claim $\mathbf{w}$, which will then be used to either confirm or deny its veracity. More formally, given $\mathbf{w}$ and a set of related documents $\mathbf{d}$, a transformer-based model denoted $\mathrm{VC}_{\mathrm{Check}}$ performs the task of *stance detection* on each piece of evidence $d \in \mathbf{d}$.

[2] `https://idir.uta.edu/claimbuster`

Stance detection is formalized as a classification task, where the relationship between $d \in \mathbf{d}$ and $\mathbf{w}$ is categorized into one of 4 classes: agree, discuss, disagree, and unrelated. See Table 1 for additional details and examples.

Table 1: Stance Detection Task Definition and Examples

Class	Definition	Sample Evidence for: "Unemployment rate is at an all-time low"
Agree	d supports $\mathbf{w}$	At 2.4%, unemployment is the lowest it's been in our nation's history.
Discuss	Factuality of $\mathbf{w}$ unclear	The unemployment rate is unknown at this time.
Disagree	d refutes $\mathbf{w}$	As low as it is today, the unemployment rate was lower in 1936.
Unrelated	d is unrelated to $\mathbf{w}$	Chicken nuggets are Charles Barkley's all-time favorite food.

Given a method of performing stance detection, we now consider the challenge of selecting $\mathbf{d}$. We design `VeriClaim` to query 3 distinct resources to retrieve documents related to $\mathbf{w}$, which will later be passed to $\mathrm{VC}_{\mathrm{Check}}$ to verify the veracity of $\mathbf{w}$:

- **Knowledge Bases**: `VeriClaim` will query trusted knowledge bases such as Wolfram Alpha to retrieve specific factual information regarding $\mathbf{w}$.
- **Fact-Check Repositories**: To take advantage of the work already done by human fact-checkers, `VeriClaim` will search trusted fact-check repositories for existing reports related to $\mathbf{w}$.
- **Internet Resources**: A curated list of trusted internet resources is queried to discover *unstructured* evidence related to $\mathbf{w}$.

We also define an auxiliary Relevancy Score RS to quantify the relevance between $\mathbf{w}$ and $\mathbf{d}$:

$$RS = 1 - p(y = \text{Unrelated} \mid \mathbf{w}, \mathbf{d}; \theta_{\mathrm{VC}_{\mathrm{Check}}}) \tag{1}$$

2.2.1 Stance Detection Transformer

We first discuss $\mathrm{VC}_{\mathrm{Check}}$, since its formulation is a necessary prerequisite for processing all 3 types of evidence. Each time we pass a claim-evidence pair $\mathbf{x} = (\mathbf{w}, \mathbf{d})$ into the model, it goes through the following components:

- **Documentation-Level Tokenization**: Many articles are too long to pass into the transformer at once. [3] To address this concern, we split $\mathbf{d}$ up into components where each has a length upper-bounded by the architectural limit. This effectively expands the input into a "dataset" $\mathcal{X} = \{(\mathbf{w}, \mathbf{c}) \mid \mathbf{c} \in \mathbf{d}\}$. Each pair is passed into the transformer separately, and results are aggregated at the end.
- **Embedding Process**: Next, each input pair $(\mathbf{w}, \mathbf{c}) \in \mathcal{X}$ is embedded into an n-dimensional space. As seen in Figure 2, embeddings for $\mathbf{w}$ and $\mathbf{d}$ are separated by the embedding for the [SEP] token to indicate the boundary between a claim and its supporting evidence.
- **LM Transformer + Dense Layer**: The embedded input is then transformed into a latent representation via a sequence of {self-attention + feed-forward} layers, which is finally passed through a fully-connected layer to extract the final classification. Each dataset element $\mathcal{X}_i$ gets an output in $\mathbb{R}^4$, where each logit represents a stance detection class.
- **Aggregation**: Finally, the tokenized portions of $\mathcal{X}$ are recombined via averaging to give a final stance prediction. The end-to-end stance detection pipeline can be expressed concisely as:

$$\texttt{stance\_detect}(\mathbf{x}) = f(\mathcal{X} = \{(\mathbf{w}, \mathbf{c}) \mid \mathbf{c} \in \mathbf{d}\}; \boldsymbol{\theta}) \tag{2}$$

$$= \left\{ \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{w},\mathbf{c}) \in \mathcal{X}} \mathrm{VC}_{\mathrm{Check}}(\mathbf{w}, \mathbf{c}; \boldsymbol{\theta})_j \;\middle|\; 0 \leq j < 4 \right\} \tag{3}$$

To train $\mathrm{VC}_{\mathrm{Check}}$, we utilize the Fake News Challenge (FNC) stance detection dataset, which is described in additional detail in Section 3.

[3] 512 token is a common upper limit on input sequence length.

2.2.2 Evidence Retrieval & Evidence-Level Verdicts

In this subsection, we outline `VeriClaim`'s evidence retrieval scheme, which consults 3 trusted categories of resources to find evidence $\mathbf{d}$ that may confirm, debunk, or support discussion of $\mathbf{w}$.

Knowledge Bases: The Wolfram Alpha knowledge base serves as a reliable and constantly-updated source for knowledge. Responses from the Wolfram knowledge base is conveniently given in raw-text form. However, given a claim $\mathbf{w}$, we must consider the challenge of converting $\mathbf{w}$ into a query. Since Wolfram accepts natural-language queries, we turn to Question Generation (QG) models to transform $\mathbf{w}$ into an interrogative question $\mathbf{q}$. See Table 2 for examples of Claim-Question pairs.

Table 2: Examples of Claim-Question Pairs

Claim	Sample Question
Barack Obama was the 10th President of the US.	Who was the 10th President of the US?
Each year, millions die from homicide in the US.	How many people die from homicide in the US per year?
Vaccines cause autism.	What causes autism?

Neural models have recently exhibited state-of-the-art performance on QG tasks [4, 5]. Particularly, Microsoft Research's Unified Language Model (UniLM) Framework [5] has achieved top results on SQuAD-QG [6]. Thus, we employ UniLM to generate high-quality $\mathbf{q}$ from $\mathbf{w}$.

Not all interrogative questions generated by UniLM will be suitable for querying Wolfram Alpha, so we employ Wolfram's Fast Query Recognition API [4] to filter out unsuitable queries. After the question pool is deemed clean, Wolfram's Spoken Results API [5] is used to generate natural-language responses to all $\mathbf{q}$. See Figure 3 for a visualization of this process.

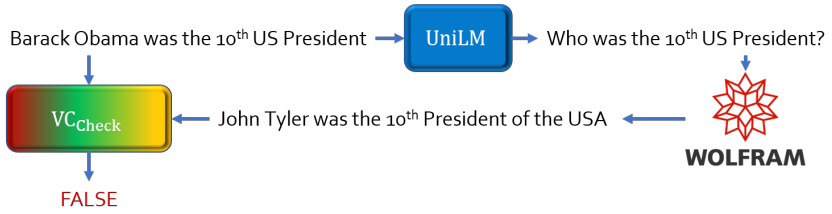


Figure 3: Knowledge Base Query Process

Applying VC_{Check} to the claim-evidence pairs from this stage returns a matrix $R^{kb} \in \mathbb{R}^{N_{kb} \times 4}$, where each row represents a piece of evidence and the 4 columns correspond to the 4 stance detection classes.

Fact-Check Repositories: To select additional evidence, `VeriClaim` also searches for existing fact-check reports from reputable institutions including Politifact, Snopes, and Fullfact. To execute such queries, we apply the Google Fact Check Explorer (FCE) API [6], which returns a list of relevant claims given each input, as well as the truth verdicts assigned to them by professional fact-checkers.

However, determining the relationship between these "related fact-checks" (i.e. pieces of evidence) and the original claim is not trivial; we propose a two-step process to solve this. First, VC_{Check} is applied to evaluate agreement and relevance between the input claim and each related claim; this returns a matrix $R^{repo} \in \mathbb{R}^{N_{repo} \times 4}$, again with each row representing one claim-evidence pair and each column representing a class. Then, for each related claim, if the associated verdict is `false`, we swap the agree and disagree scores from VC_{Check}.

Internet Resources: Finally, to take advantage of unstructured data, we build a pipeline to extract relevant evidence from Internet articles. We use the Media Bias Fact Check (MBFC) Repository, which contains 2,500 websites manually-annotated for factuality, to set up 4 Google Custom Search API [7] Engines:

[4] `http://products.wolframalpha.com/query-recognizer/documentation/`
[5] `http://products.wolframalpha.com/spoken-results-api/documentation/`
[6] `https://toolbox.google.com/factcheck/apis`
[7] `https://developers.google.com/custom-search/v1/overview`

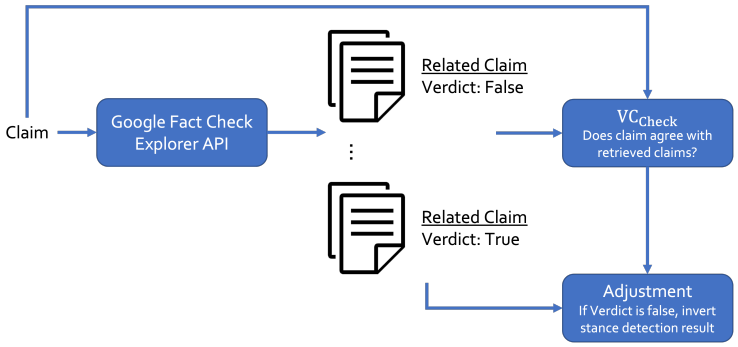


Figure 4: Fact-Check Repository Query Process

- High Factuality: Scientific analysis, low political bias
- Medium Factuality: Slight left/right political bias
- Low Factuality: Conspiracy/rumor websites
- Wikipedia Corpus: English Wikipedia encyclopedia entries

A query executor is used to retrieve relevant websites from some combination of the 4 search engines (by default, only *High Factuality* is queried). Then, given a set of URLs, we employ a web scraper to extract HTML content, which is then post-processed using Trafilatura [7] to remove boilerplate.

Passing this evidence into VC_{Check} returns a matrix $R^{inet} \in \mathbb{R}^{N_{inet} \times 4}$ (same formatting as the previous two evidence categories).

2.3 Scoring Formula

To aggregate the results from each piece of evidence, we define a *relevance-weighted* scoring formula, which takes into account the confidences of VC_{Check} and RS for each piece of evidence. Its formulation is given in Equation 4.

$$\begin{cases} S_{agree} = \frac{1}{\sum_{t \in \mathcal{S}} N^t} \sum_{t \in \mathcal{S}} \sum_{i=1}^{N_t} R^t_{i,0} \times (1 - R^t_{i,3}) \\ S_{discuss} = \frac{1}{\sum_{t \in \mathcal{S}} N^t} \sum_{t \in \mathcal{S}} \sum_{i=1}^{N_t} R^t_{i,1} \times (1 - R^t_{i,3}) \\ S_{disagree} = \frac{1}{\sum_{t \in \mathcal{S}} N^t} \sum_{t \in \mathcal{S}} \sum_{i=1}^{N_t} R^t_{i,2} \times (1 - R^t_{i,3}) \end{cases} \quad \text{where} \quad \mathcal{S} = \{kb, repo, inet\} \tag{4}$$

Note that the scores defined in Equation 4 are not normalized. Therefore, to define the final normalized scores S_{n_c} for each class c such that $\sum S_{n_c} = 1$, normalization is performed as: $S_{n_c} = \frac{S_c}{\sum S_i}$.

3 Model-Based Evaluation

3.1 Claim-Spotting Model Analysis

We refer the reader to *Gradient-Based Adversarial Training on Transformer Networks for Detecting Check-Worthy Factual Claims* [3] for full results on the claim-spotter.

3.2 Claim-Checking Model Analysis

We evaluate VC_{Check} on the Fake-News Challenge (FNC) [8] dataset, which consists of claim-document pairs that are binned into 4 classes, as discussed in Section 2.2. FNC consists of 75,000 total datapoints, with 49,972 training and 25,028 testing samples. [9] The baseline and state-of-the-art model comparisons are shown below.

[8] `http://www.fakenewschallenge.org/`

[9] It isn't obvious that claim-checking should be formulated as a stance detection task, but the FNC hosts held extensive interviews with fact-checkers and concluded this.

Table 3: Stance Detection Classification Results

Model	Agree	Discuss	Disagree	Unrelated	Acc	F1-Mac
Gradient Boosting	14.8	2.0	69.5	96.5	86.3	46.1
SOLAT in the SWEN [10]	53.8	3.6	76.0	97.9	89.1	57.8
Memory Network [8]	—	—	—	—	88.6	56.9
Fakta [9, 10]	54.6	15.1	72.6	97.7	88.2	60.0
Multi-Task Learning [11]	—	—	—	—	92.3	74.4
VC_{Check}	56.4	34.6	74.5	97.6	88.6	65.8

As the above table shows, VC_{Check} outperforms all but one past work: *Neural Multi-Task Learning for Stance Prediction* [11]. While there are certainly methods to train larger language transformers, as [11] did, we opted for a smaller BERT Base model, which improved inference speed by a factor of ≥ 2. Inspired by [11], however, we are interested in integrating multi-task pre-training and larger (but efficient) transformer models into `VeriClaim`.

4 Real-World Fact-Checking Performance

4.1 Dataset Creation

Dataset performance is only one part of the equation. What about the combination of our evidence retrieval and stance detection methods *in the wild*? To evaluate this, we curate a dataset of 29 claims made by Joe Biden, Donald Trump, and Hillary Clinton, the three most recent Presidential candidates. [11] The objective is for `VeriClaim` to score as many correctly as possible.

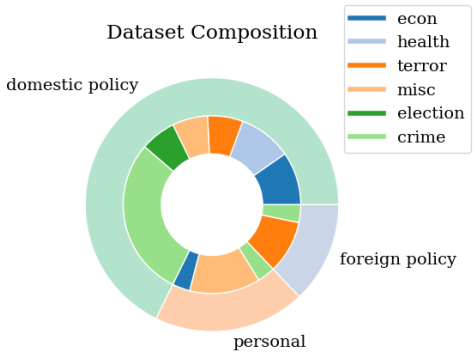


Figure 5: Custom Dataset Topic Composition

These 29 claims (14 true and 15 false) were collected from PolitiFact's website's fact check list page. For some entity $e = \{$`<FIRST_NAME>`, `<LAST_NAME>`$\}$, we can find the claims made by e with factuality `<FACTUALITY>` at the url `www.politifact.com/factchecks/list/?speaker=<FIRST_NAME>-<LAST_NAME>&ruling=<FACTUALITY>`.

4.2 Results & Analysis

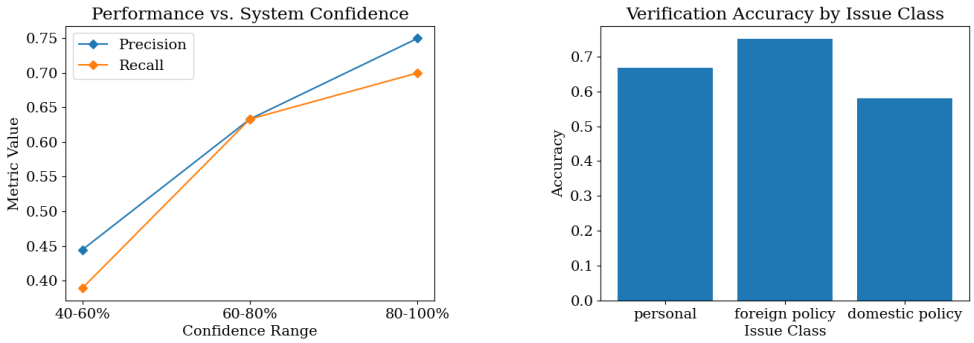
Table 4 displays the results of running `VeriClaim` on the aforementioned dataset. Moreover, we analyze `VeriClaim`'s performance with respect to two variables: confidence in its decision (Figure 6-a) and the category of issue (Figure 6-b).

Notice that, in Table 4, recall on false statements is significantly higher than that of true statements. This indicates the degree to which `VeriClaim` views information critically; it would rather categorize a true claim as false than a false claim as true. This is arguably better than the alternative; the cost on missing out on a piece of false information is, in our opinion, higher than the opposite. In the latter case, humans would simply need to read through `VeriClaim`'s info dashboard (Section 5.1) and conclude for themselves that the information agrees with the claim.

[11] As of the 2020 Election.

Table 4: Fact Verification Results

Class	Precision	Recall	F1-Score
True Statement	0.70	0.50	0.58
False Statement	0.61	0.73	0.67
Macro Average	0.66	61.5	0.63



(a) Confidence vs. Performance

(b) Issue Class vs. Performance

Figure 6: Performance Analysis

Regarding Figure 6, we find significance in `VeriClaim`'s confidence scores: the more confident the probabilities, the more likely a decision is to be correct. This indicates adequate *calibration* of the system. Moreover, Figure 6-b shows that performance is fairly uniformly distributed across fact domains, which aligns with our expectations.

4.3 Observations on Failure Cases

One particularly striking failure case is with *numerical inputs*. Each of the 29 dataset entries was labelled as either numerical or non-numerical in nature, and out of all the incorrect predictions, 72.7% came from claims centered around a number. A few are shown in Table 5.

Table 5: Examples of Incorrectly Classified Quantitative Claims

Poverty rates for African Americans have reached their lowest levels in the history of our country.
We spend more per student than almost any other major country in the world.
A shocking 20 veterans are committing suicide each and every day, especially our older veterans.

We attribute these failures to the emphasis on language semantics, rather than quantitative reasoning, in present-day neural networks; large language models, even ones as sizable as GPT-3, have been shown to struggle with this [12]. It seems reasonable that, to grapple with the nuances of quantitative claims, we need to investigate more explicit symbolic reasoning algorithms.

5 Demo Applications

In this section, we cover methods to more smoothly communicate fact-checkers to users.

5.1 Online Fact-Checking Interface

Here, users can input a large block of text that they would like to both claim-spot and claim-check. The interface supports content-rich displays for in-depth factuality analysis. In particular, users can review the specific documents that `VeriClaim`'s claim-checker used to formulate its verdict. Each evidential document can be clicked to reveal a detailed analysis page, which includes a paragraph-by-paragraph breakdown.

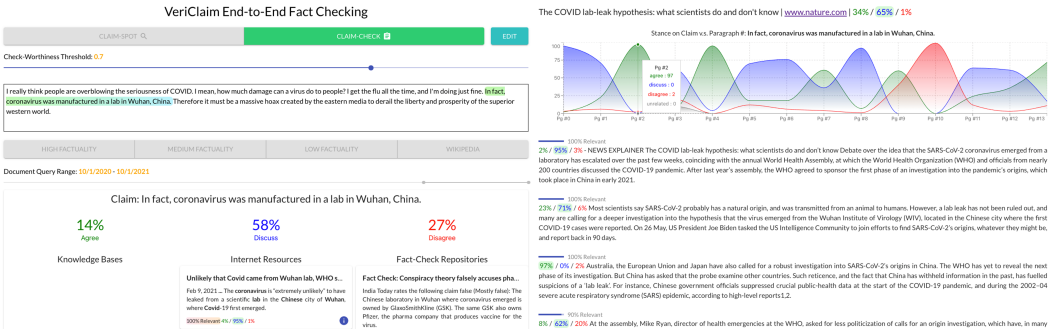


Figure 7: `VeriClaim` Website Dashboard and Detailed Breakdown

5.2 Chrome Browser Extension

Next, we implement a Google Chrome Browser Extension to bring fact-check reports directly to the user. The extension 1) scrapes user-visited webpages for their main contents, 2) performs claim-spotting, and 3) aggregates claim-checking results over all check-worthy claims to produce an overall factuality report. This information is all displayed in an extension popup, which provides hyperlinks for users who want more detailed analysis.

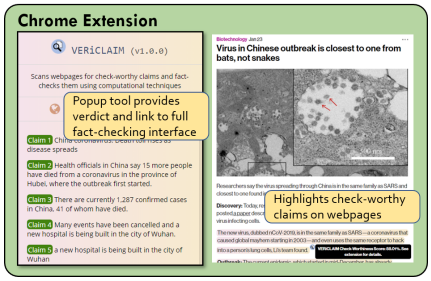


Figure 8: `VeriClaim` Chrome Extension

6 Related Works, Conclusion, and Discussion

Over the years, people have built a variety of systems for automatic end-to-end factual monitoring and verification. These include ClaimBuster [2], Fakta [10], and WikiCheck [13]. While ClaimBuster's primary focus is identifying claims that are worth checking, Fakta and WikiCheck focus exclusively on checking claims *after* they've been selected, and also with undue biases (only selecting from news articles or Wikipedia, for example).

`VeriClaim` improves upon these past works by increasing the diversity of its evidence retrieval methods, as well as streamlining the entire fact-checking process. Needless to say, however, **computational fact-checking is hard**. This remains far from being a "solved problem"; even from an evaluative standpoint, it's difficult to find papers that analyze real-world performance like we do. Fakta is evaluated purely on FNC, WikiCheck on FEVER [14], and ClaimBuster not at all.

While deep learning will certainly be a part of top-performing systems at least for the near future, we strongly believe that future solutions can benefit from more careful, targeted engineering. Language is full of nuance, and it's unreasonable to expect our present-day neural networks to generalize universally. What if we built algorithms that could carefully dissect certain classes of claims (and be very good at this), rather than lean on large-brush, general-purpose systems that become jacks of all trades yet masters of none?

Ruminations about deep learning aside, what `VeriClaim` also brings is a focus on *communicating* fact-checks better to the general public, which is a critical yet easily overlooked component of social impact research. It is our hope that researchers and fact-checkers alike can ultimately benefit from this work.

7 References

[1] G. Pennycook and D. G. Rand, "Fighting misinformation on social media using crowdsourced judgments of news source quality," *Proceedings of the National Academy of Sciences*, vol. 116, no. 7, pp. 2521–2526, 2019.

[2] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, V. Sable, C. Li, and M. Tremayne, "Claimbuster: The first-ever end-to-end fact-checking system," *Proceedings of the VLDB Endowment*, vol. 10, pp. 1945–1948, Aug. 2017.

[3] K. Meng, D. Jimenez, F. Arslan, J. D. Devasier, D. Obembe, and C. Li, "Gradient-based adversarial training on transformer networks for detecting check-worthy factual claims," *arXiv:2002.07725*, 2020.

[4] M. Chinkina and D. Meurers, "Question generation for language learning: From ensuring texts are read to supporting learning," in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 334–344, Association for Computational Linguistics, Sept. 2017.

[5] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *EMNLP*, 2016.

[7] A. Barbaresi, "Trafilatura: A web scraping library and command-line tool for text discovery and extraction," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, (Online), pp. 122–131, Association for Computational Linguistics, Aug. 2021.

[8] M. Mohtarami, R. Baly, J. Glass, P. Nakov, L. Màrquez, and A. Moschitti, "Automatic stance detection using end-to-end memory networks," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, June 2018.

[9] B. Xu, "Combating fake news with adversarial domain adaptation and neural models," in *NeurIPS Workshop on Stance Detection, Fact Checking, Adversarial Domain Adaptation*, 2019.

[10] M. Nadeem, W. Fang, B. Xu, M. Mohtarami, and J. Glass, "FAKTA: An automatic end-to-end fact checking system," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, June 2019.

[11] W. Fang, M. Nadeem, M. Mohtarami, and J. Glass, "Neural multi-task learning for stance prediction," in *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, (Hong Kong, China), pp. 13–19, Association for Computational Linguistics, Nov. 2019.

[12] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," *arXiv preprint arXiv:2103.03874*, 2021.

[13] M. Trokhymovych and D. Saez-Trumper, "Wikicheck: An end-to-end open source automatic fact-checking api based on wikipedia," *arXiv preprint arXiv:2109.00835*, 2021.

[14] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and VERification," in *NAACL-HLT*, 2018.