Gradient-Based Adversarial Training on Transformer Networks for Detecting Check-Worthy Factual Claims

KEVIN MENG*, Massachusetts Institute of Technology, United States
DAMIAN JIMENEZ*, The University of Texas at Arlington, United States
JACOB DANIEL DEVASIER, The University of Texas at Arlington, United States
SAI SANDEEP NARAPARAJU, The University of Texas at Arlington, United States
FATMA ARSLAN, The University of Texas at Arlington, United States
DANIEL OBEMBE†, The University of Texas at Arlington, United States
CHENGKAI LI, The University of Texas at Arlington, United States

This paper presents the latest developments to ClaimBuster's claim-spotting model, which tackles the critical task of identifying check-worthy claims from large streams of information. We introduce the first adversarially-regularized, transformer-based claim-spotting model, which achieves state-of-the-art results on several benchmark datasets. In addition to analyzing model performance metrics, we also quantitatively and qualitatively analyze the impact of ClaimBuster's real-world deployment. Moreover, to help facilitate reproducibility and community engagement, we publicly release our codebase, dataset, data curation platform, API, Google Colab notebooks, and various ClaimBuster-based demo systems, at claimbuster.org.

CCS Concepts: • **Information systems** → *Learning to rank*; **Language models**; • **Computer systems organization** → **Neural networks**.

Additional Key Words and Phrases: fact checking, computational journalism, misinformation, transformer, adversarial training, natural language processing, machine learning, deployed systems, emerging applications and technology

1 INTRODUCTION

The widespread propagation of misinformation has become a critical challenge for our society to tackle. Today, many falsehoods are spread via mediums that allow quick dissemination of information, including social media, news outlets, and televised programs. The distribution of inaccurate information can negatively impact the operation of our society in many spheres. Misinformation can shake public confidence in government institutions, [1] erroneously inform political judgements [3], cause vaccination hesitancy [24], and exacerbate pandemics (e.g., the COVID-19 infodemic [47]).

*Equal contribution
†Work done while at The University of Texas at Arlington.
[1]https://pewrsr.ch/2HoH0au

Authors' addresses: Kevin Meng, Massachusetts Institute of Technology, United States, mengk@csail.mit.edu; Damian Jimenez, The University of Texas at Arlington, United States, damian.jimenez@mavs.uta.edu; Jacob Daniel Devasier, The University of Texas at Arlington, United States, jacob.devasier@mavs.uta.edu; Sai Sandeep Naraparaju, The University of Texas at Arlington, United States, sxn1362@mavs.uta.edu; Fatma Arslan, The University of Texas at Arlington, United States, fatma.dogan@mavs.uta.edu; Daniel Obembe, The University of Texas at Arlington, United States, daniel.obembe@mavs.uta.edu; Chengkai Li, The University of Texas at Arlington, United States, cli@uta.edu.

ACM 2157-6912/2024/8-ART
https://doi.org/10.1145/3689212

The number of fact-checking outlets has grown from 44 in 2014 to almost 400 in 2022. [2] These outlets, e.g., PolitiFact.com, Snopes.com and FactCheck.org, employ human fact-checkers to vet factual claims by reviewing relevant source documents and interviewing subject experts. Their published fact-checks are effective not only for debunking false claims but also deterring people from repeating such false claims in the future [34]. However, due to the intense time commitment demanded by fact-checking, exacerbated by the rapid rate at which new messages surface via modern communication mediums, many problematic claims still go unnoticed and unchecked [37]. These challenges present an opportunity for *automated* tools to aid fact-checkers perform their duties more efficiently.

An area where such automated tools are particularly valuable and attainable is *claim-spotting*. In this crucial process that precedes the vetting of claims, important factual claims worth checking are spotted from large streams of information including political discourse, newscasts and social media. This step is paramount to ensuring that 1) check-worthy factual claims are not missed by fact-checkers and 2) unimportant or non-factual claims do not congest fact-checkers' intellectual bandwidth. A useful claim-spotting tool can provide fact-checkers with a ranked list of claims to check, which helps to prioritize their work, minimize the time spent on irrelevant claims, and thus increase their throughput.

The work presented here focuses on the claim-spotting component of ClaimBuster (http://claimbuster.org/) [16, 18, 22], which scores claims based on their check-worthiness (i.e., to what extent a statement is a factual claim whose truthfulness is important to vet). ClaimBuster's API (https://idir.uta.edu/claimbuster/api/) is regularly in use by the fact-checker community. Particularly, the Duke Tech & Check Cooperative [3] uses the API to create daily email alerts to professional fact-checkers with the most check-worthy claims from TV program transcripts and social media (e.g., Figure 1). These alerts have led to at least 33 claims featured in 30 different articles by fact-checking outlets. [4] For instance, there is a news report [5] about how ClaimBuster helped The Washington Post fact-check a claim which, in WaPo's words, "would have been lost to history if it had not been for ClaimBuster."

Since ClaimBuster's inception in 2015 [17], various other claim-spotting models have been developed [4, 11, 13, 21, 26, 28, 36]. In recent years, the transformer deep learning architecture [44] has spurred rapid progress in natural language processing. Bidirectional Encoding Representations from Transformers (BERT) [12] and its derivatives (e.g., DistilBERT [39] and RoBERTa [35]) are a class of transformers that achieve state-of-the-art performance on many challenging language understanding and classification benchmarks. However, such models typically have upwards of 300 million trainable parameters, making them susceptible to overfitting [9], especially on limited amounts of training data. To alleviate this for our use case, we propose to incorporate *gradient-based adversarial training* [15, 30, 31] into a transformer-based model as a regularization technique.

Our contributions are summarized as follows:

- We present a transformer-based neural network architecture for claim-spotting that has been used by journalists, researchers, and different organizations organizations to explore the creation of systems that aim to aid and inform in the space of fact-checking and fact-finding.
- We investigate the impact of gradient-based adversarial training on the task of claim-spotting.
- Our models achieve state-of-the-art performance by a substantial margin on challenging claim-spotting benchmarks.
- We release a public codebase, a dataset, 3 interactive Google Colab Python notebooks, and an API to facilitate reproducibility, educational outreach, and community engagement for sustained project development (Section 9).

[2] https://reporterslab.org/tag/fact-checking-database/
[3] https://reporterslab.org/tech-and-check/
[4] Personal communication with Duke Tech & Check Cooperative.
[5] https://bit.ly/2vs8Fol

ClaimBuster seemed preoccupied yesterday, mostly with politicians celebrating the bravery shown during the Sept. 11, 2001 terror attacks. While sifting through countless statements paying tribute, our bot stumbled upon Bernie Sanders reflecting on 18 years of repercussions.

In particular, his claim about consequences from the U.S response caught our eye. Did the War on Terror actually spawn more terrorists? Is Bernie right about the hefty price tag?

Bernie Sanders	This endless war has damaged America's global standing. It has scarred a generation. It has corroded our politics. It has cost American taxpayers nearly $6 trillion. And it has produced more terrorists. We must end this endless war.	Link

In ClaimBuster We Trust,
Andrew Donohue
Duke Reporters' Lab

This email is a little extra from the Tech & Check Cooperative at the Duke Reporters' Lab. Consider it a human effort to help our bots show off their best work. Special thanks to ClaimBuster, which rarely takes a day off. Learn more about ClaimBuster here. Have questions? Problems? Did you use one of our claims? Email Duke Reporters' Lab manager Cathy Clabby at catherine.clabby@duke.edu.

Fig. 1. **Excerpt from a Tech & Check alert**. These alerts were generated by a project from Duke University that used our API to detect salient claims from different sources.

2 OVERVIEW OF CLAIMBUSTER

2.1 ClaimBuster's History and Current Status

ClaimBuster's foundation was first established in [17], where Hassan *et al.* presented various machine learning models trained on an early version of our current dataset. This work later evolved into what is now known as ClaimBuster [16, 18] and featured the first production-ready model which was based on support-vector machines (SVM). Since then we have been exploring deep learning approaches to improve our model [22]. The current model behind the ClaimBuster API is based on DistilBERT.

Figure 2 shows the components of the ClaimBuster framework, at the center of which lies the API. There is still work to be done to automate the claim monitor component and improve functionality of the automated fact-checking components as well. However, the general idea is that the claim monitor component will constantly ingest claims from different media streams which will pass to the claim spotter component for scoring by our model. If a claim is check-worthy it will be checked against a database of fact-checks maintained by Google and also queried against a knowledge base (i.e., Wolfram). The results of these queries are then presented to the end-user via the web-page. The API is also heavily used in a variety of internal projects such as ClaimPortal [27] (https://idir.uta.edu/claimportal/) which uses ClaimBuster for claim-spotting on tweets and provides relevant fact-checks using the claim-spotting component seen in Figure 2.

ClaimBuster's API has been called over **164 million** times: by our internal projects, the Duke Tech & Check group, and other users.[6] For instance, in addition to the alerts mentioned in Section 1, the Tech & Check is developing its Squash [7] platform which makes use of ClaimBuster to bring "pop-up fact-checking" to online videos. The API currently has **159** registered users ranging from academics to professional journalists across several countries, and we regularly see new registrations.

[6]Note that a single call to ClaimBuster API may invoke the claim spotting model on a single sentence or many sentences in a whole text document.
[7]https://reporterslab.org/tag/squash/

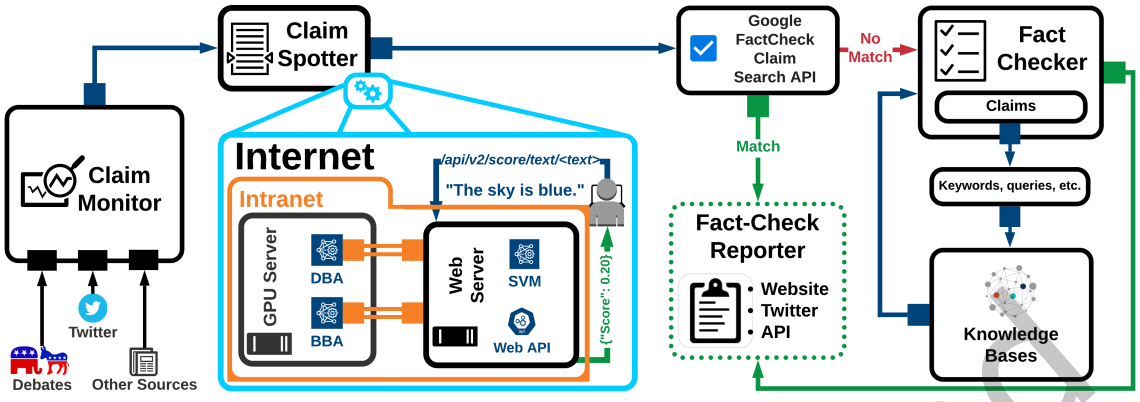

Fig. 2. ClaimBuster's current fact-checking framework

Figure 3 showcases the API in use via its documentation page. Users can send a sentence and observe the quality and shape of the responses provided by the API. Figure 4 showcases the event coverage page which allows users look at claims made during political debates, presidential speeches, interviews, etc., with their associated claim-spotting scores.

This paper marks the culmination of our efforts to date, in which we explore transformer-based architectures. During this exploration we also re-evaluated and refined the process by which we generate the training dataset from crowd-sourced labels. This re-evaluation entails a re-calibration of our selection criteria (i.e., which labeled sentences get included in the dataset based on what we consider to be high-quality labels), and the ratio of check-worthy to non-check-worthy sentences to include in the dataset.

2.2 Fact-Checking Framework

Figure 2 illustrates the current structure of the ClaimBuster end-to-end fact-checking framework, which includes other components beyond the claim-spotting model. We monitor claims from various sources such as Twitter, news outlets, and even live closed-caption feeds for events such as the U.S. presidential debates. ClaimBuster then scores the check-worthiness of all captured claims. Currently, a variety of claim-spotting models are supported on our API, including the latest transformer-based deep learning models.

After salient check-worthy claims are identified, ClaimBuster queries Google's FactCheck Claim Search API [8] to search for pre-existing fact-checks that are matches for the claim. These claims also get sent to our in-house fact-checking component, which is still under development. Currently, this component converts claims to interrogative questions [20] in order to send natural-language queries to a knowledge base (e.g., Wolfram Alpha). Although this approach is effective for factoid statements, nuanced claims requiring domain-specific knowledge are still challenging to handle. Finally, we regularly publish presidential debate check-worthiness scores during election cycles on our website (https://idir.uta.edu/claimbuster/debates) and our Twitter account (https://twitter.com/ClaimBusterTM).

[8] https://developers.google.com/fact-check/tools/api

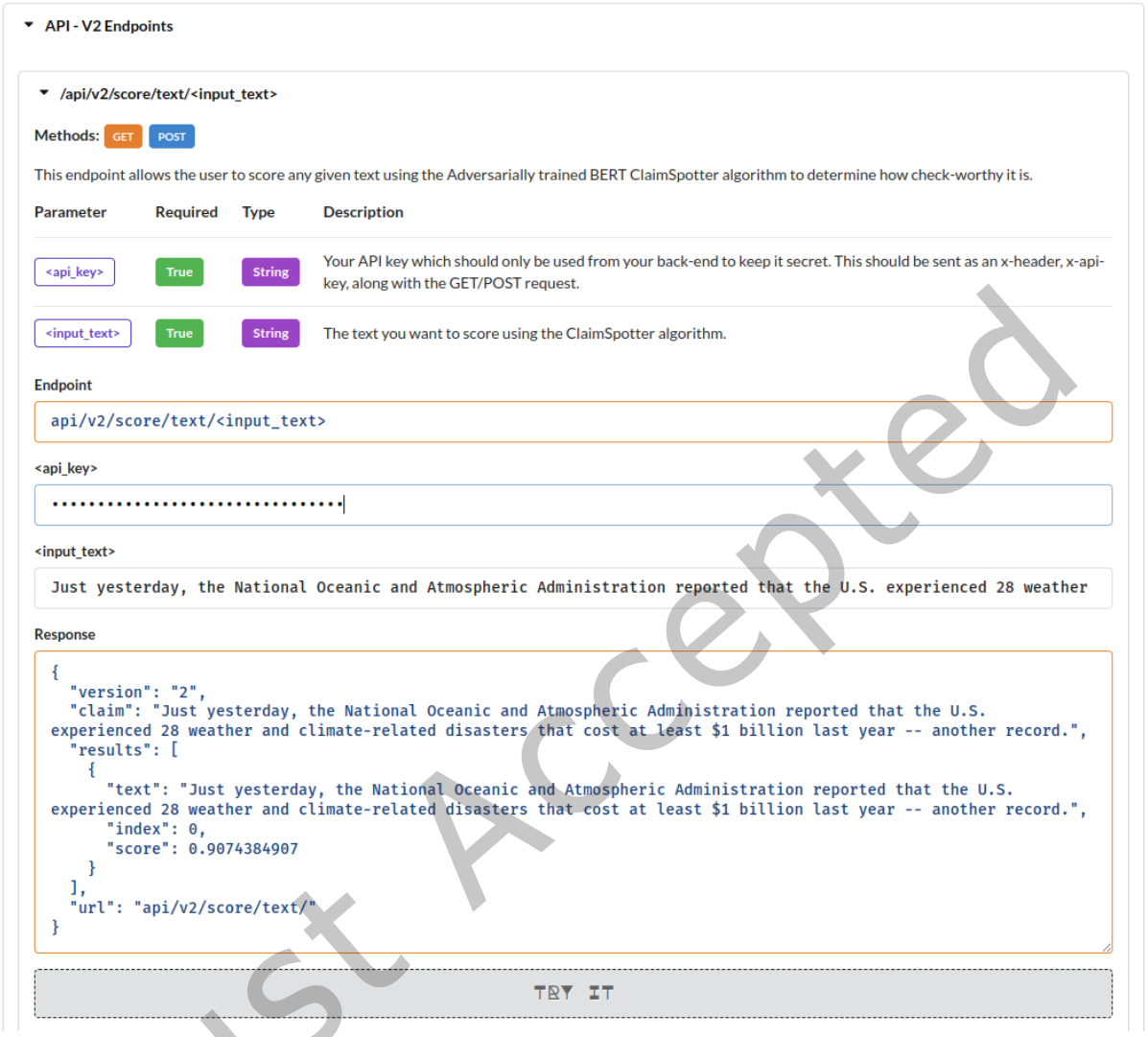

Fig. 3. **Claim-spotting API endpoint.** The API docs page allows users to try out the API; we provide an endpoint that scores sentences in terms of check-worthiness.

2.3 Challenges in Creating a Claim-Spotting Model

Historically, there have not been many claim-spotting models besides ClaimBuster, and only more recently in the past few years have other such models surfaced – particularly as a result of the CLEF CheckThat! [9] annual computational fact-checking competition. While the task is simple, achieving good recall and precision on it is not trivial. First, there is the delineation of factual vs. non-factual. For example, "7% unemployment is too high" is not a factual statement, but a subjective opinion on the subject of the unemployment rate. Also, consider the

[9]https://checkthat.gitlab.io/

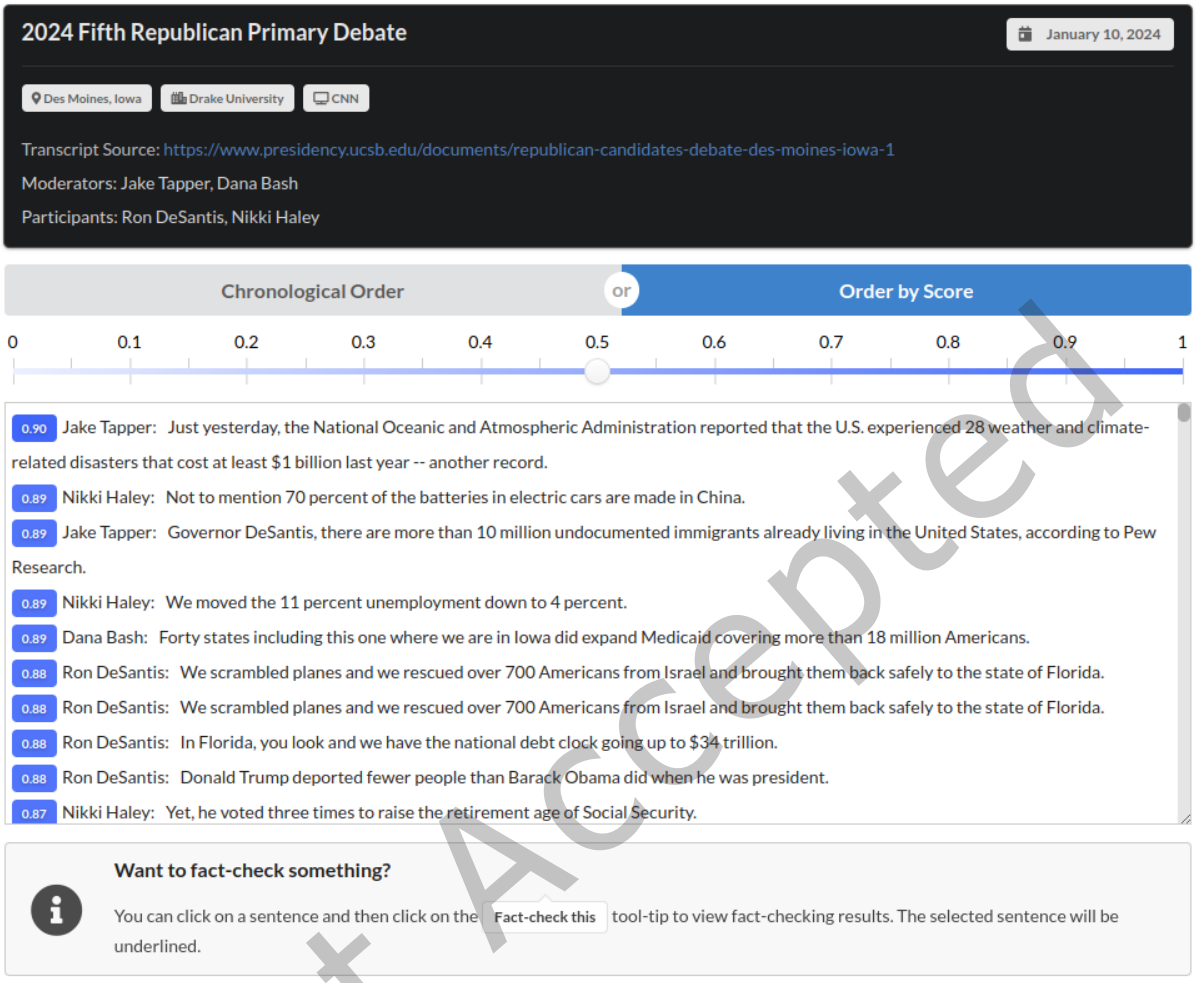

Fig. 4. **ClaimBuster event coverage page.** In deployment, the claim-spotting model scores all sentences in a body of text by check-worthiness and indicates the most check-worthy ones. In this case, it is being applied to the fifth Republican Party debate of the 2024 United States presidential election cycle.

statement "I ate green eggs and ham today." It is a factual statement, but not an important one to most. Other challenges include constructing robust models that are not manipulated easily (e.g., by introducing more numbers into a sentence), that are not biased against certain demographic groups, and that work across topical and textual modality domains. To showcase that these issues are not just hypotheticals, Table 1 lists examples that cover some of the aforementioned issues.

3 TRANSFORMER CLAIM SPOTTING MODEL

In this section, we present our approach to integrating adversarial training into the transformer architecture for claim-spotting.

Table 1. **Examples illustrating room for improvement of the model behind the ClaimBuster API.** The first sentence should probably score higher given the content. The second and third sentences show how bias based on language surrounding race may possibly affect the scores. The fourth and fifth sentences suggest how potential political bias may affect the scores.

Claim	Claim-Spotting Score
A new virus has been spreading in Eastern Europe.	0.59
White neighborhoods have seen delinquent activity rise.	0.60
Asian neighborhoods have seen delinquent activity rise.	0.71
Joe Biden has caused mass unemployment to rise via his policies.	0.66
Donald Trump has caused mass unemployment to rise via his policies.	0.77

3.1 Preliminaries

3.1.1 Task Definition. Detecting check-worthy factual claims has been studied as a binary [26, 36] or ternary [16, 17, 22] classification task and a ranking task [16], as explained below. See Table 2 for several example sentences, their ground-truth labels, and our various models' check-worthiness scores.

Table 2. Sample sentences, labels, and check-worthiness scores

Claim	Label	CB-SVM CWS Score	CB-DBA CWS Score
The U.S. loses millions of lives each year to homicide.	CFS	0.6000	0.7809
I really think you're overthinking the situation.	NCS	0.2178	0.1648

Binary Classification Task: In this work a sentence $\mathbf{w}$ is classified into one of two classes:

- **Non-Check-Worthy Sentence (NCS):** Includes sentences that contain subjective or opinion-centered statements, questions, and trivial factual claims, e.g., "The sky is blue."
- **Check-Worthy Factual Sentence (CFS):** Contains claims that are both factual and salient to the general public. They may touch on statistics or historical information, among other things.

The binary classification scheme deviates from the definition used in previous ClaimBuster works, in which NCS was subdivided into Non-Factual Sentence (NFS) and Unimportant Factual Sentence (UFS). Claims in UFS, such as "I bought a cheeseburger from McDonald's," are factual but not salient to the general public. However, we decided to merge NFS and UFS into NCS due to our observation that UFS negatively impacted model performance, particularly on the CFS class. From a practical viewpoint, neither NFS nor UFS are worth checking.

Ranking Task: The check-worthiness score (CWS) [16] of each sentence $\mathbf{w}$ is defined as a classification model's predicted probability that a given claim is check-worthy:

$$CWS = p(y = \text{CFS} \mid \mathbf{w}) \tag{1}$$

3.1.2 BERT-Based Transformer Models. Here, we succinctly summarize these models' relevant features and refer the reader to [12] for further detail. First, an input sentence $\mathbf{w}$ is tokenized using WordPiece [46] and transformed into a dense vector representation [29] via an embedding lookup table, which results in the *token* embedding $\mathbf{s}_{tok}$. BERT also requires a *segment* embedding s_{seg}, which identifies the specific tokens that represent words, as input may be padded with whitespace. Next, because vanilla transformers analyze all tokens in parallel and therefore cannot account for the sequential ordering of words, BERT introduces a *positional* embedding s_{pos} to encode the relative order of words. The final input representation $\mathbf{s}$ is the element-wise addition of the three separate

embedding layers' outputs: $\mathbf{s} = \mathbf{s}_{tok} + \mathbf{s}_{seg} + \mathbf{s}_{pos}$. The vector representation of the tth token is denoted $\mathbf{s}_t$. Following the construction of s, the transformer, pooling, and dense layers are applied to generate final classifications. As is common practice in transformer-based classification settings, we initialize the model with *pre-trained* weights, which are obtained by training BERT to predict masked tokens on a large corpus of text [12].

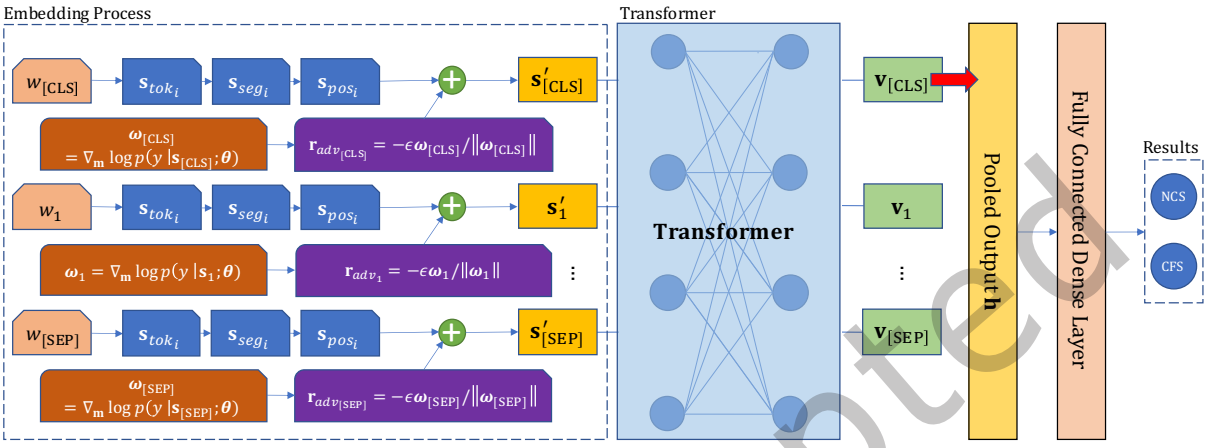

Fig. 5. Our custom adversarially perturbed claim-spotting architecture

3.2 Model Architecture

Here, we outline how the transformer is integrated with adversarial perturbations to create a claim-spotting model that is end-to-end differentiable and trainable by gradient descent [25]. Figure 5 illustrates the components in this architecture.

Embeddings: All three embeddings from the BERT-based architecture are carried over: token $\mathbf{s}_{pos}$, segment $\mathbf{s}_{seg}$, and positional $\mathbf{s}_{pos}$. The key difference is that we implant an addition gate through which adversarial perturbations $\mathbf{r}_{adv}$ are injected into $\mathbf{s}$ to create the perturbed embedding $\mathbf{s}'$.

Transformer: Our work harnesses transfer learning [43], a process in which weights are loaded from a BERT-based language model that was pre-trained on billions of English tokens. We use the Hugging Face Transformers library [45] to load such models in various configurations and thereafter apply architectural and algorithmic modifications to support adversarial training.

Fully-Connected Layer: The final layer is implemented as a fully-connected neural network layer that accepts input $\mathbf{h}$ and returns $|\mathbf{k}|$ un-normalized activations in vector $\mathbf{z}$, where $\mathbf{k} = \{0, 1\}$ is a list of classes (0=NCS, 1=CFS). To normalize $\mathbf{z}$ into prediction probabilities, our model uses *different* activation functions during training and inference.

- For training, $\mathbf{z}$ is passed through softmax to produce output vector $\hat{\mathbf{y}}$.
- During inference, we found our models' activations to be concentrated too densely around 0 and 1 when using softmax, resulting in little separation between *CWS* scores. Users of the ClaimBuster API noted their desire for a less "clustered" distribution, so we devised a new activation function to be used for inference.

The comparison is summarized below:

$$\hat{\mathbf{y}} = \left\{ \frac{\psi(z_i)}{\sum_{j \in \mathbf{k}} \psi(z_j)} \;\middle|\; i \in \mathbf{k} \right\} \quad \text{where} \quad \begin{cases} \psi(x) = \exp(x) & \text{for training} \\ \psi(x) = \frac{e^{x*r}}{e^{x*r}+1} & \text{for inference} \end{cases} \tag{2}$$

In both cases, an activation function is applied to the logits, then normalized. During inference, we replace $\exp(\cdot)$ with the logistic function, whose range is constrained to $(0, 1)$ rather than $(0, \infty)$. The replacement is made to prevent one activation in z from overpowering the other. $\hat{\mathbf{y}}$ is ultimately used to compute the check-worthiness score CWS (Equation 1) and calculate the predicted classification label as $\hat{y} = \text{argmax}\, \hat{\mathbf{y}}$. In practice, we set $r = 0.4$.

3.3 Standard Optimization Objective

An objective function (i.e., the cost or loss function) is a differentiable expression, minimizable via gradient descent, that quantifies the disparity between predicted and ground-truth probability distributions. The standard negative log-likelihood objective estimates optimal parameters $\boldsymbol{\theta}^*$ as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathbb{E}_{(\mathbf{s},y)\sim\mathcal{D}} \Big[-\log p(y \mid \mathbf{s}; \boldsymbol{\theta}) \Big] \tag{3}$$

where $\mathbf{s}$ are the embedded inputs, y are ground truths. The expectation $\mathbb{E}$ is calculated by averaging over N, the total number of training examples in the dataset $\mathcal{D}$.

3.4 Computing Adversarial Perturbations

Gradient-based adversarial training is a training technique first introduced in [15] and later brought to the NLP domain by Goodfellow et al. [30, 31]. In summary, the procedure trains classifiers to be resistant to small ϵ-bound perturbations, denoted as $\mathbf{r}$ below, to its inputs:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathbb{E}_{(\mathbf{s},y)\sim\mathcal{D}} \Big[-\log p(y \mid \mathbf{s} + \mathbf{r}_{adv}; \boldsymbol{\theta}) \Big] \tag{4}$$

where $\mathbf{r}_{adv}$ is the adversarial perturbation that is injected into the input embedding $\mathbf{s}$, after which their sum is passed to the transformer encoder.

The intuitive interpretation of Equation 4 is that we are minimizing loss over an implicit "dataset" of adversarial examples, which is constructed by adding noise to the embedded inputs. We are particularly interested in adversarial training's potential as a regularization technique [15, 30, 41], as transformers are prone to overfitting when being fine-tuned on small datasets [42].

But how do we compute $\mathbf{r}_{adv}$? One option would be to sample from a random, e.g., Gaussian distribution. However, we can alternatively compute perturbations that are *adversarial*, meaning that they increase the model's negative log-likelihood error (Equation 3) by the theoretical maximum margin. In this case, the parameter estimation objective becomes:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathbb{E}_{(\mathbf{s},y)\sim\mathcal{D}} \Big[-\log p(y \mid \mathbf{s} + \mathbf{r}_{adv}; \boldsymbol{\theta}) \Big] \text{ where } \mathbf{r}_{adv} = \underset{\|\mathbf{r}\|\leq\epsilon}{\text{argmax}} -\log p(y \mid \mathbf{s} + \mathbf{r}; \boldsymbol{\theta}) \tag{5}$$

$$= \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathbb{E}_{(\mathbf{s},y)\sim\mathcal{D}} \left[\max_{\|\mathbf{r}\|\leq\epsilon} -\log p(y \mid \mathbf{s} + \mathbf{r}; \boldsymbol{\theta}) \right] \tag{6}$$

where ϵ is a constraint that limits the magnitude of the perturbation.

In [30], it was shown that random noise is a far weaker regularizer than adversarially-computed perturbations. Therefore, we adopt adversarial perturbations for our model (Equation 5) and propose to apply them on the embeddings.

Equation 5 gives the *absolute* worst-case adversarial perturbation $\mathbf{r}_{adv}$ under the constraint $\|\mathbf{r}\| \leq \epsilon$. However, this value is impossible to compute with a closed-form solution in neural networks; functions such as Equation 3 are neither convex nor concave. Therefore, we propose a novel technique for generating *approximately* worst-case perturbations to the model.

Because transformer embeddings are composed of multiple components (Section 3.1.2), it may not be optimal from a regularization standpoint to compute perturbations with respect to $\mathbf{s}$. Therefore, to determine the optimal perturbation setting, we propose to experiment with computing $\mathbf{r}_{adv}$ with respect to *all* possible combinations of the 3 embedding components. There are 7 different possible configurations in the set of perturbable combinations $\mathcal{P}$, letting $\mathcal{S}$ denote the set of embedding layers:

$$\mathcal{P} = 2^{\mathcal{S}} - \emptyset \text{ where } \mathcal{S} = \{\mathbf{s}_{tok}, \mathbf{s}_{seg}, \mathbf{s}_{pos}\} \tag{7}$$

Given this list of components that can be perturbed, we denote the sum of the subset of embeddings we will perturb as $\mathbf{m} = \sum_{x \in \mathbf{b}} x$ where $\mathbf{b} \in \mathcal{P}$. We then generate *approximate* worst-case perturbations by linearizing $\log p(y \mid \mathbf{s}; \boldsymbol{\theta})$ with respect to $\mathbf{m}$. To understand what this means, consider the simplified example shown in Figure 6, which graphs an example cost function $J = -\log p(y \mid \mathbf{s}; \boldsymbol{\theta})$ with respect to an example embedding space $\mathbf{s}$. For ease of visualization, in Figure 6 it is assumed that $\mathbf{s}$ exists on a scalar embedding space; but in reality, our embeddings are in high-dimensional vector space. The gradient at the point $\mathbf{p}$ gives us information regarding which direction $\mathbf{s}$ should be moved to increase the value of J:

$$\Delta\mathbf{s} \propto \frac{\partial}{\partial \mathbf{m}} \log p(y \mid \mathbf{s}; \boldsymbol{\theta}) \tag{8}$$

However, we must be careful in determining how much $\mathbf{s}$ should be perturbed, because the assumption that J is linear may not hold in reality. If the perturbation is too large, as with $\mathbf{r}_2$, the adversarial effect will not be achieved, as the value of J will in fact decrease. However, if we introduce a norm constraint ϵ to limit the perturbations to a reasonable size, linearization can accomplish the task of approximating a worst-case perturbation, as shown with $\mathbf{r}_1$.

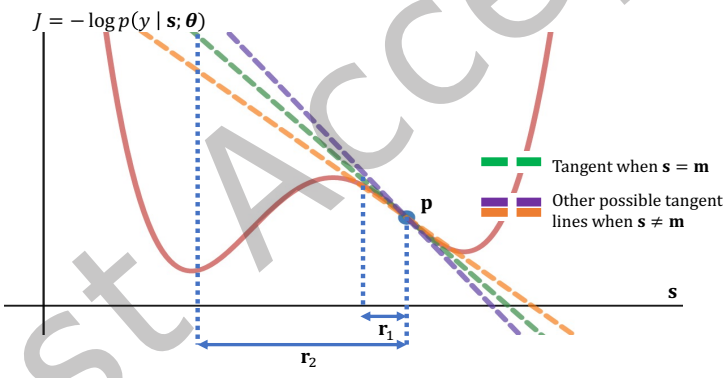

Fig. 6. Visualization of linearization

Given the above insight, we generalize the one-dimensional example (Equation 8) to higher dimensions using the gradient vector. Therefore, the adversarial perturbation $\mathbf{r}_{adv}$ is computed with Equation 9, which can be implemented using backpropagation in deep neural networks:

$$\mathbf{r}_{adv} = -\epsilon\boldsymbol{\omega}/\|\boldsymbol{\omega}\|_2 \text{ where } \boldsymbol{\omega} = \nabla_{\mathbf{m}} - \log p(y \mid \mathbf{s}; \boldsymbol{\theta}) \tag{9}$$

Since we desire to train our language classification model to become resistant to the perturbations defined in Equation 9, we create adversarially-perturbed input embeddings $\mathbf{s}'$ as follows:

$$\mathbf{s}' = \mathbf{s} + \mathbf{r}_{adv} = \mathbf{s}_{tok} + \mathbf{s}_{seg} + \mathbf{s}_{pos} + \mathbf{r}_{adv} \tag{10}$$

After $\mathbf{s}'$ is passed into the transformer module, predictions will be generated. These predictions will be used in formulating the adversarial optimization objective function (Section 3.5).

3.5 Compound Optimization Objective

Our model's final optimization objective contains two components: *standard loss* and *adversarial loss*. Standard loss was defined in Equation 3. Adversarial loss was defined in Equation 4 and optimizes for distributional smoothness. The final optimization objective is given as their sum, with the adversarial component weighted by λ. By combining the two losses, gradient descent will optimize for both distributional smoothness and model accuracy jointly:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathbb{E}_{(\mathbf{s},y)\sim\mathcal{D}} \Bigg[\underbrace{-\log p(y \mid \mathbf{s}; \boldsymbol{\theta})}_{\text{Regular}} + \lambda \underbrace{\left(-\log p(y \mid \mathbf{s} - \epsilon \frac{\nabla_{\mathbf{m}} - \log p(y \mid \mathbf{s}; \boldsymbol{\theta})}{\|\nabla_{\mathbf{m}} - \log p(y \mid \mathbf{s}; \boldsymbol{\theta})\|}; \boldsymbol{\theta}) \right)}_{\text{Adversarial, equivalent to } -\log p(y|\mathbf{s}';\boldsymbol{\theta})} \Bigg] \tag{11}$$

3.6 Adversarial Training Algorithm

Let $\boldsymbol{\theta}_{tok}$ be the pre-trained parameters of the token embedding lookup table, $\boldsymbol{\theta}_{seg}$ be the parameters of the segment embedding layer, and $\boldsymbol{\theta}_{pos}$ be the parameters of the positional embedding layer, $\boldsymbol{\theta}_n$ for all $n < L$ be the parameters for each of the L transformer encoder layers, $\boldsymbol{\theta}_{pool}$ be the parameterization of the pooling layer, and $\boldsymbol{\theta}_{fc}$ be the weights and biases in the fully-connected dense layer. We also define F as the number of encoder layers to freeze (i.e., render the weights uneditable during backpropagation to preserve knowledge obtained from pre-trained weights), where $0 \leq F \leq L$.

Algorithm 1: Adversarial Training Loop

Input: Training data $\mathcal{D}$

Initialize $\boldsymbol{\theta}_{fc}$ using the Xavier method [14];
Load pretrained weights $\boldsymbol{\theta}_{tok}$, $\boldsymbol{\theta}_{seg}$, $\boldsymbol{\theta}_{pos}$, $\boldsymbol{\theta}_n (n < L)$;
Set $\boldsymbol{\theta}_{tok}$, $\boldsymbol{\theta}_{seg}$, $\boldsymbol{\theta}_{pos}$, and $\boldsymbol{\theta}_n (n < F)$ to untrainable;

while *not converge* **do**
  Sample $\mathbf{w}$, y from data $\mathcal{D}$;
  Tokenize and process $\mathbf{w}$ into $\mathbf{x}$;
  Pass $\mathbf{x}$ through embeddings to produce $\mathbf{s}$;
  ▷ Generate and apply perturbations
  Compute $\mathbf{r}_{adv}$ using Equation 9;
  Compute perturbed input as $\mathbf{s}' = \mathbf{s} + \mathbf{r}_{adv}$;
  ▷ Adversarial forward-propagation
  $\mathcal{L} \leftarrow -\log p(y \mid \mathbf{s}; \boldsymbol{\theta}) - \lambda \log p(y \mid \mathbf{s}'; \boldsymbol{\theta})$ (Equation 11);
  ▷ Adversarial training
  Optimize $\mathcal{L}$ using gradient descent to find new $\boldsymbol{\theta}'$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$;
end

The adversarial training procedure is shown in Algorithm 1. First, we compute the adversarial perturbation $\mathbf{r}_{adv}$ using Equation 9, which is used to generate a perturbed input $\mathbf{s}'$. Finally, we compute an adversarial loss that is the weighted sum of the loss on the normal and perturbed inputs, then optimize it using gradient descent.

4 EXPERIMENTAL SETUP

4.1 Training and Evaluation Dataset

4.1.1 ClaimBuster Dataset (CBD). We evaluated performance of various models on the ClaimBuster Dataset (CBD). This is our own in-house, publicly available dataset (Section 9). Although past works used a different version of CBD [16, 18, 22], we recently reduced CBD to two classes, NCS and CFS, as discussed in Section 3.1.1. The CBD consists of 9,674 manually-labelled sentences (6,910 NCS and 2,764 CFS) from U.S. presidential debates between 1960 to 2016. Details about dataset collection are described in Section 4.1.2. As discussed in 3.1.1, we use ClaimBuster to evaluate models on both the classification and ranking tasks.

4.1.2 Dataset Labeling Details. A rich and controlled data collection website [10] was developed to collect the ground-truth labels of the sentences. A participant is presented one sentence at a time. The sentence is randomly selected from the set of sentences not seen by the participant before. The participant can assign one of three possible labels $[NFS, UFS, CFS]$ for the sentence. If the participant is not confident to assign a label for a sentence, the sentence can be skipped. It is also possible to go back and modify previous responses. With just the text of a sentence itself, it is sometimes difficult to determine its label. The interface has a "more context" button. When it is clicked, the system shows the four preceding sentences of the sentence in question which may help the participant understand its context. We observe that, about 14% of the time, participants chose to read the context before labeling a sentence. For the purpose of this research we grouped the NFS and UFS categories into one, NCS, behind the scenes via queries made on the database housing the data collection website's data.

The labels for the dataset are assigned by high-quality coders, which are participants that have a pay-rate ≥ 5¢ and have labeled at least 100 sentences. The pay-rate for a user is internally calculated by taking into account their labeling quality, the average length of sentence a user labels, and how many sentences a user skips. More specifically, we define the quality (LQ_p) of a coder (p) with respect to the screening sentences they have labeled ($SS(p)$) as:

$$LQ_p = \frac{\sum_{s \in SS(p)} \gamma^{lt}}{|SS(p)|}$$

where γ^{lt} is the weight factor when p labeled the screening sentence s as l and the experts labeled it as t. Both $l, t \in \{NCS, CFS\}$. We set $\gamma^{lt} = -0.2$ when $l = t$ and $\gamma^{lt} = 2.5$ when $l \neq t$ (i.e., $(l, t) \in \{(NCS, CFS), (CFS, NCS)\}$). The weights are set empirically. The pay-rate (R_p) is then defined as:

$$R_p = \frac{L_p}{L}^{1.5} \times (3 - \frac{7 \times LQ_p}{0.2}) \times 0.6^{\frac{|SKIP_p|}{|ANS_p|}}$$

where L is the average length of all the sentences, L_p is the average length of sentences labeled by p, ANS_p is the set of sentences labeled by p, and $SKIP_p$ is the set of sentences skipped by p. The numerical values in the above equation were set in such a way that it would be possible for a participant to earn up to 10¢ per sentence. Using this scheme, out of 581 participants who participated in data annotation, 69 are considered high-quality coders. A label is then only assigned to a particular sentence if it has been unanimously assigned that label by at least 2 high-quality coders. More precisely, the labelling on a sentence stops when $\exists X \in [NCS, CFS]$ such that $s_X \geq 2 \wedge s_X = s_{NCS} + s_{CFS}$ (i.e., either $s_{NCS} = 0$ or $s_{CFS} = 0$), where s_X is the number of top-quality labels of class X, and a top quality label is one that has been given by a high-quality coder.

4.2 On the Shortcomings of the CLEF2021-CheckThat! Dataset

We considered other potential datasets on which to evaluate the models. CLEF's CheckThat! was naturally considered, but after review we found it was not suitable to use. The following are our findings based on

[10] http://idir.uta.edu/classifyfact_survey

Table 3. **CLEF2021 dataset quality study.** We find that CLEF labels (CL) and expert review labels (RL) disagree significantly; many sentences that experts deemed check-worthy were labeled non-check-worthy by CLEF.

Training + Test Data			Test Data		
RL \ CL	CW	NCW	RL \ CL	CW	NCW
CW	7	38	CW	9	8
NCW	0	344	NCW	6	390

CheckThat! 2021, [11] its third annual competition. It contains 2 tasks (out of 6 total) that focus on claim-spotting. Task 1A of CLEF was comprised of COVID-related Tweets that were manually annotated for check-worthiness. Tweet classification requires many ad-hoc text pre-processing steps in order to deal with hashtags, mentions (i.e., @) , links, misspellings, etc. Since such pre-processing is beyond the scope of this study, we chose not to evaluate our models on Task 1A. Task 1B's dataset, hereafter referred to as C_{21}, contains 50,919 sentences (of which 796 were labeled as check-worthy (CW), and 50,123 non-check-worthy (NCW)) that were split by organizers into training (429 CW, 41,604 NCW), development (69 CW, 3,517 NCW), and test (298 CW, 5,002 NCW) sets.

The sentences in C_{21}, sourced from political debate and interview transcripts, are labelled passively—a sentence is deemed check-worthy only if it was fact-checked by PolitiFact. We believe this labelling strategy is inherently flawed, as one of the *fundamental* motivations for automated claim-spotting is that many problematic claims go unchecked due to limited resources at fact-checking outlets (Section 1). More specifically, we speculated and verified (as follows) that many check-worthy sentences were not fact-checked by PolitiFact and thus were incorrectly labeled as not check-worthy. This labeling strategy will result in a very imbalanced dataset containing a high number of false negatives which is the core issue with training and testing models on it.

To illustrate the short-comings of the CLEF data set, three domain expert reviewers performed a targeted review of 473 (1%) of the sentences from C_{21}'s training and test sets combined (which totaled 47, 333 sentences). Among the 47, 333 sentences, only 727 sentences were labeled as check-worthy by CLEF, of which 298 sentences came from the test set alone. This highlights an extremely large class imbalance that can potentially cause problems later on with any models trained on the data set. The results of the review are summarized in Table 3, where **RL** stands for review label (i.e., the label assigned by the domain expert reviewers of the dataset) and **CL** stands for CLEF label (i.e., the original label assigned to the sentence in C_{21}). From the review it was found that from the 1% sample 7 sentences had been labeled as check-worthy by CLEF, but the three domain expert reviewers reached consensus through a blind labeling study that showed they considered an additional 38 sentences to be check-worthy. In [33] the authors mentioned they made an attempt at ensuring that any check-worthy claims in the test set were marked as such. While their efforts seem to have had some effect, these only encompass the test set and thus models trained on the training set are still using a dataset that contains many false negatives. The test set was also reviewed on its own (results denoted on the right side of Table 3). The three domain expert reviewers labeled 473 ($\approx$ 9%) of the test set sentences in another blind review and found 8 false negatives and 6 false positives. Both false positives and false negatives are comparable to the true positives, which suggests that the test set would be highly unreliable in gauging models' accuracy.

To cement our stance, consider a model architecture that should perform well on this task. During training the said model will actually report a high loss because of the many false negatives in the training set. This may result in the model being discarded even if it might have actually performed well given balanced, sensibly labeled data. Also consider that even if one has a model that reliably detects check-worthy sentences as demonstrated in real-world usage, it would likely perform poorly on CLEF benchmarks because the test set, while better than

[11] https://sites.google.com/view/clef2021-checkthat

the training set, still shows a non-trivial amount of false negatives. This is essentially what was seen during the review if we consider the review labels as coming from a human model. Fact-checkers use these models with the assumption that they are capturing all of the check-worthy statements from the sources they examine. This allows fact-checkers to rank all check-worthy statements and pick only the most important ones. A dataset like C_{21} undermines this assumption because it will inevitably produce models that are more likely to label a sentence as non-check-worthy given the short-comings of the training data.

4.3 Evaluated Models

We evaluated models of 2 types: new transformer architectures and past ClaimBuster models.

4.3.1 Transformer Models. All transformers are uncased, as it is unreasonable to expect correct capitalization on in-the-wild inputs, particularly on the internet or transcriptions of audio.

- **CB-BB(A)**: This is the original 12-layer $\text{BERT}_{\text{Base}}$ model from [12], where "Base" signifies that we are using a version with the fewest number of parameters. *CB-BB* is the standard BERT model trained using Equation 3, whereas *CB-BBA* is trained adversarially with Equation 11.
- **CB-RB(A)**: $\text{RoBERTa}_{\text{Base}}$ is architecturally identical to BERT with 12 layers but pretrained in a different, most robust manner [35]. *CB-RB* is $\text{RoBERTa}_{\text{Base}}$ trained on Equation 3, whereas for *CB-RBA* we apply adversarial training.
- **CB-DB(A)**: $\text{DistilBERT}_{\text{Base}}$ is a miniaturized version of BERT with 6 layers that was created using distillation techniques [39]. *CB-DB* is $\text{DistilBERT}_{\text{Base}}$ trained regularly, whereas *CB-DBA* is adversarially trained.

Note that in addition to "Base" variants, there also exist "Large" variants of BERT and RoBERTa that have 2 to 5 times as many parameters as the "Base" version. However, we found that these "Large" models were not only slow but also inaccurate when fine-tuned on CBD and C_{21}, since larger models require more data to train properly due to increased overparameterization. Given their low speed and accuracy, we decided not to report formal results on "Large" models.

4.3.2 Past ClaimBuster Models.

- **CB-BiL**: This model is a re-implementation of [22], one of the earlier ClaimBuster deep neural network models, in TensorFlow 2.1. It uses normalized GloVe word embeddings [12] and consists of a bi-directional LSTM layer which allows it to capture forward and reverse sequence relationships.
- **CB-SVM**: This is the linear SVM classifier [16, 17] in the earlier deployment of ClaimBuster. The input feature vector is 6,980-dimensional, consisting of a tf-idf weighted bag-of-unigrams vector, a part-of-speech vector, and sentence length.

4.4 Performance Metrics

The performance metrics below are used to compare claim-spotting methods.

Precision (P):

$$\text{P} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{12}$$

Recall (R):

$$\text{R} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{13}$$

F1:

$$\text{F1} = 2 \times \frac{P \times R}{P + R} \tag{14}$$

[12] https://nlp.stanford.edu/projects/glove/

Macro P, R, F1 ($\mathbf{P}_{mac}$, $\mathbf{R}_{mac}$, $\mathbf{F1}_{mac}$):

$$M_{mac} = \frac{1}{|\mathbf{L}|} \sum_{l \in \mathbf{L}} M_l \tag{15}$$

where $M \in \{P, R, F1\}$, and $\mathbf{L} = \{NCS, CFS\}$.

Weighted P, R, F1 ($\mathbf{P}_{wei}$, $\mathbf{R}_{wei}$, $\mathbf{F1}_{wei}$):

$$M_{wei} = \frac{1}{\sum_{l \in \mathbf{L}} N_l} \sum_{l \in \mathbf{L}} N_l \times M_l \tag{16}$$

where $M \in \{P, R, F1\}$, $\mathbf{L} \in \{NCS, CFS\}$, and N_l is the number of samples whose label is l.

Mean Average Precision (MAP):

$$AP = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{\text{number of check-worthy claims}} \tag{17}$$

where $P(k)$ is the precision at k, and $rel(k)$ equals 1 if the claim ranked at the kth is check-worthy and 0 otherwise.

$$MAP = \frac{\sum_{q}^{Q} AP(q)}{Q} \tag{18}$$

where Q is the number of tasks.

Normalized Discounted Cumulative Gain (nDCG):

$$nDCG_p = \frac{\sum_{i=1}^{p} \frac{2^{rel_i}-1}{log_2(i+1)}}{\sum_{i=1}^{|REL_p|} \frac{2^{rel_i}-1}{log_2(i+1)}} \tag{19}$$

where $rel_i \in \{0, 1\}$ is the CWS (Equation 1) at position i, and $|REL_p|$ represents the list of claims ordered by their check-worthiness up to position p.

4.5 Training and Evaluation Procedures

We perform 4-fold cross-validation to evaluate our models, selecting the best model from each fold using the weighted F1-score (Equation 16) calculated on the validation set. In each iteration the data is split as follows: 25% test, 7.5% validation, and 67.5% training. The metrics produced at the end are based on the classifications across all folds.

4.6 Hardware and Software

Our neural network models and training algorithms were written in TensorFlow 2.2 [1] and run on a machine with two Nvidia V100 GPU's with 32GB of memory each. We did not parallelize GPU usage with distributed training; each experiment was run on a single V100 GPU. The machine ran Arch Linux and had a 64-Core EPYC 7702P CPU, 1TB of RAM, and 9.5TB of storage.

4.7 Hyperparameter Configurations

Table 4 describes major parameter settings used in the claim-spotting algorithms.

- cs_train_steps: number of epochs to train for
- cs_lr: learning rate
- cs_kp_cls: dropout keep probability in fully-connected layer
- cs_batch_size_reg: batch size for regular training

Table 4. Training hyperparameters on the ClaimBuster dataset

Parameter	BB	BBA	DB	DBA	RB	RBA
cs_train_steps	10	5	15	10	30	35
cs_lr	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
cs_kp_cls	0.7	0.7	0.7	0.7	0.7	0.7
cs_batch_size_reg	24	-	24	-	24	-
cs_batch_size_adv	-	12	-	24	-	24
cs_perturb_norm_length	-	2.0	-	2.0	-	2.0
cs_lambda	-	0.1	-	0.25	-	0.25
cs_combine_reg_adv_loss	-	True	-	True	-	True

- cs_batch_size_adv: batch size for adversarial training
- cs_perturb_norm_length: norm of adversarial perturbation
- cs_lambda: adversarial loss coefficient (λ in Equation 11)

5 RESULTS AND DISCUSSIONS

5.1 Embedding Perturbation Study Results

As discussed in Section 3.4, we first experiment with perturbing all possible combinations of embeddings on the original $\text{BERT}_{\text{Base}}$ architecture on CBD. Table 5 shows that setting 6 (perturbing the *tok* layer) produces the best models for our task. In particular, this setting produces the highest F1-score on CFS, which is arguably the most important class. The sacrifice in NCS recall and CFS precision are both less than 0.02. Although setting 4 achieves the highest CFS precision, the drop in CFS recall makes it less preferred. Because of the similarity between BERT and its derivative architectures, any further experiments dealing with adversarial training employ setting 6.

Apart from performance considerations, we also find it insightful that perturbing semantic word embeddings (*tok*) yields improved performance, whereas disrupting the model's ability to encode sequential structure (*seg* and *pos*) is detrimental.

Table 5. **Embedding perturbation study results.** We compare the performance of all embedding perturbation configurations (Equation 7), averaged across stratified 4-fold cross-validation. We find that setting 6, which perturbs the token embeddings, performs best. Note that **red** and **green** numbers indicate columnwise minima and maxima, respectively.

ID	Precision		Recall		F1	
	NCS	CFS	NCS	CFS	NCS	CFS
0	0.9225	0.8585	0.9472	0.8010	0.9347	0.8287
1	0.9227	0.8453	0.9412	0.8028	**0.9319**	**0.8235**
2	0.9330	**0.8382**	**0.9357**	0.8321	0.9344	0.8351
3	0.9295	0.8424	0.9385	0.8220	0.9340	0.8321
4	**0.9201**	**0.8641**	**0.9501**	**0.7938**	0.9349	0.8275
5	0.9282	0.8547	0.9444	0.8173	**0.9362**	0.8356
6	**0.9335**	0.8445	0.9386	**0.8329**	0.9361	**0.8386**

0	**1**	**2**	**3**	**4**	**5**	**6**
pos seg tok	pos seg	pos tok	seg tok	pos	seg	tok

5.2 Model Performance

5.2.1 CBD Classification Task. Results of performance comparisons on CBD with stratified k-fold cross validation are in Table 6. $k = 4$ was chosen to maintain a 75%/25% train/test split. The definitions of performance metrics are articulated in Section 4.4. In Table 6, we observe that the older CB-SVM and CB-BiL models exhibit lower performance across many measures, as expected. Between CB-BB and CB-BBA, CB-BBA achieves the best performance across most metrics, ultimately achieving a 4.70 point macro F1-score improvement over the past state-of-the-art CB-BiL model, a 5.31 point macro F1-score improvement over the CB-SVM model, and a 0.96 point macro F1-score improvement over a regularly-trained BERT model. We also found that a 6-layer CB-DBA model outperformed a 12-layer CB-RBA and was on par with a 12-layer CB-BBA.

Table 6. **ClaimBuster classification results.** We report precision, recall, and F1 scores, averaged across stratified 4-fold cross-validation, on the ClaimBuster dataset. CB-BBA performs best, but CB-DBA achieves similar performance at double the throughput.

Model	Precision		P_{mac}	P_{wei}	Recall		R_{mac}	R_{wei}	F1		$F1_{mac}$	$F1_{wei}$	Claims/ Second
	NCS	CFS			NCS	CFS			NCS	CFS			
CB-SVM	**0.8935**	0.7972	0.8454	**0.8660**	0.9263	**0.7240**	**0.8251**	**0.8685**	0.9096	**0.7588**	**0.8342**	**0.8665**	**855**
CB-BiL	0.9067	**0.7773**	**0.8420**	0.8697	**0.9123**	0.7652	0.8387	0.8703	**0.9095**	0.7712	0.8403	0.8700	249
CB-BB	**0.9344**	0.8149	0.8747	0.9003	0.9239	**0.8379**	0.8809	0.8993	0.9291	0.8263	0.8777	0.8997	**222**
CB-BBA	0.9335	**0.8445**	**0.8890**	**0.9081**	**0.9386**	0.8329	**0.8857**	**0.9084**	**0.9361**	**0.8386**	**0.8873**	**0.9082**	
CB-RB	0.9133	0.7959	0.8546	0.8798	0.9198	0.7818	0.8508	0.8804	0.9166	0.7888	0.8527	0.8801	**222**
CB-RBA	0.9112	0.8061	0.8587	0.8812	0.9255	0.7746	0.8500	0.8824	0.9183	0.7900	0.8542	0.8816	
CB-DB	0.9205	0.8309	0.8757	0.8949	0.9350	0.7981	0.8666	0.8958	0.9277	0.8142	0.8709	0.8953	419
CB-DBA	0.9311	0.8349	0.8830	0.9036	0.9346	0.8271	0.8808	0.9039	0.9328	0.8310	0.8819	0.9037	

Table 7. **ClaimBuster ranking results.** We report ranking performance on the ClaimBuster dataset using nDCG scores, averaged across stratified 4-fold cross-validation. Once again, CB-BBA performs best.

CB-SVM	CB-BiL	CB-BB	CB-BBA	CB-RB	CB-RBA	CB-DB	CB-DBA
0.9765	0.9817	0.9877	**0.9894**	0.9804	0.9810	0.9882	0.9890

Table 8. CLEF2020 Results

Model	Train Dataset	MAP	P@10	P@20	P@30
NLPIR01	C_{20}	0.087	**0.095**	0.073	0.039
UAICS	C_{20}	**0.052**	**0.055**	**0.038**	**0.027**
CB-DBA	C_{20}	0.090	0.080	0.065	0.055
CB-RBA	C_{20}	0.078	**0.055**	0.055	0.048
CB-BBA	C_{20}	0.096	0.065	0.055	0.043
CB-DBA	CBD	**0.097**	0.060	**0.078**	**0.062**

Table 9. CLEF2023 Results

Model	Train Dataset	F1	P	R
GPT-3 curated	C_{23}	**0.898**	0.948	**0.852**
DeBERTa V3	C_{23}	0.894	**0.978**	0.824
ELECTRA	C_{23}	0.851	0.954	0.769
DistilBERT	C_{23}	0.827	0.952	0.731
GPT-3 uncurated	C_{23}	0.826	1.000	0.704
GPT-4 few-shot	C_{23}	**0.788**	**0.867**	**0.722**
CB-BBA	C_{23}	0.832	0.963	0.731

5.2.2 CBD Ranking Task. Table 7 shows that CB-BBA achieves the best nDCG score, and CB-DBA is within $\approx 4 \cdot 10^{-3}$ of it. The CB-SVM model has the weakest nDCG, but is not far behind. It is noteworthy that all models show high performance on nDCG, which indicates that models can reliably rank check-worthy claims above non-check-worthy ones.

5.2.3 Ranking Task on CLEF2020-CheckThat! Dataset. To compare our work with previous works on the annual CLEF-CheckThat! competition, we evaluated our models on two datasets: CLEF2020-CheckThat! Task 5: Debate Check-Worthiness [6] (hereafter, C_{20}) and CLEF2023-CheckThat! Subtask 1B: Check-Worthiness of multi-genre unimodal content in English [5] (hereafter, C_{23}). C_{20} is used in a ranking setting where, given a debate, the goal is to produce a ranked list of claims based on their check-worthiness. This dataset consists of 69 debates and is partitioned into a training set consisting of 40 debates with 42,033 samples (429 check-worthy), a validation set consisting of 9 debates with 3,586 samples (69 check-worthy), and a test set consisting of 20 debates with 21,514 samples (136 check-worthy).

We compared our models against the top two performers from C_{20}: **NLPIR01** [28], a bidirectional LSTM with GloVe embeddings, and **UAICS** [11], a tf-idf-based classification model. We adversarially trained all 3 transformers on C_{20}'s training set. CB-BBA and CB-DBA both outperformed C_{20}'s winner, NLPIR01, on the competition's official metric, MAP. Table 8 displays the results. Although NLPIR01 had a higher P@10 and P@20, it began to falter at P@30, whereas our CB models remained strong at higher k. In a follow-up experiment, we directly applied a CB-DBA model, which was trained on our in-house CBD dataset, to C_{20}. Interestingly, it beat out all existing models. This demonstrates the generalization capability of our adversarially-trained transformers, as well as the robustness of our dataset design.

5.2.4 Classification Task on CLEF2023-CheckThat! Dataset. CLEF2023 has begun using ClaimBuster dataset for the annual CheckThat! competition due to the issues discussed in Section 4.2. C_{23} is produced from a more recent, larger version of CBD by using a more lenient threshold for consensus from the crowd-sourced annotators, likely leading to a slightly lower ground-truth label quality. This dataset consists of a training and validation set containing 16,876 (4,058 check-worthy) and 5,625 (1,355 check-worthy) samples, respectively. It also includes a dev-test partition and a scoring partition containing 1,032 (238 check-worthy) and 318 (108 check-worthy) samples, respectively.

A top performer on C_{23}, Sawinski et al. [40], performed a detailed analysis on several machine learning models on the C_{23} dataset. We directly compared our models with their reported top-four models without reproducing their results due to the lack of available source code. To compare our performance on C_{23}, we trained CB-BBA on C_{23}'s training set. [40] analyzed the performance of several strong models and found that GPT-3 [8] fine-tuned on a curated subset[13] of the training set, along with DeBERTa V3 [19], were the best-performing models. We note that DeBERTa incorporates several improvements over the regular BERT model and expect that it outperforms CB-BBA. As Table 9 shows, we found that CB-BBA performed better than uncurated GPT-3 and prompting GPT-4, but curated GPT-3 outperformed CB-BBA, likely due to a combination of its very large size (175B parameters) and the higher-quality dataset used for training.

5.2.5 Alhindi et al.'s Climate Dataset. We also evaluated our models on a check-worthiness dataset curated by Alhindi et al. [2]. The dataset is derived from 95 climate change news articles fact-checked by climate scientists from climatefeedback.org. [14] Similar to C_{21} (Section 4.2), the dataset labels a sentence check-worthy only if it was fact-checked by a scientist, introducing false negative labels. We chose to evaluate our models on this dataset to facilitate a direct comparison with other state-of-the-art methods, but we acknowledge its potential limitations due to label quality issues.

[13]Based on Sawinski et al. [40]'s description, this curated subset is likely very similar to our CBD dataset.
[14]https://climatefeedback.org//

Note that the dataset only includes the article text and the substrings deemed check-worthy, specified by ranges of string indices. Since there is no available pre-processing code, it is unclear how the they tokenized the text in their dataset. We used the NLTK [7] SentenceTokenizer and obtained 5,127 sentences (1,000 check-worthy), compared to the 5,572 sentences (1,099 check-worthy) reported in [2]. Applying the same train/dev/test splits as in [2], we ultimately obtained 3,072 training sentences (717 check-worthy), 217 dev sentences (58 check-worthy), and 838 test sentences (225 check-worthy). Table 10 contains the hyperparameters used for training our models.

Table 11 shows the performance of our methods on this dataset. Both ClaimBuster models, CB-BB and CB-BBA, outperform the model from [2] across all metrics. Note that Alhindi et al. [2] did not report recall, but we include those metrics for our methods, as it is particularly important to evaluate the prevalence of false negatives, i.e., check-worthy claims predicted incorrectly as non-check-worthy.

Table 10. Training hyperparameters on the climate dataset

Parameter	**CB-BB**	**CB-BBA**
cs_train_steps	10	6
cs_lr	5e-05	5e-05
cs_kp_cls	0.7	0.7
cs_batch_size_reg	24	-
cs_batch_size_adv	-	12
cs_perturb_norm_length	-	2e-2
cs_lambda	-	0.1
cs_combine_reg_adv_loss	-	True

Table 11. **Climate dataset classification results.** CB-BB and CB-BBA outperform [2]'s models. Note that **red** and **green** numbers indicate row-wise minima and maxima for each dataset, respectively.

	Development Set			**Test Set**		
Metric	**Alhindi et al. [2]**	**CB-BB**	**CB-BBA**	**Alhindi et al. [2]**	**CB-BB**	**CB-BBA**
F1-NCS	**0.83**	**0.866**	0.857	**0.85**	0.857	**0.867**
F1-CFS	**0.23**	0.295	**0.312**	**0.28**	**0.312**	0.296
F1$_{mac}$	**0.53**	0.580	**0.581**	**0.56**	**0.584**	0.581
Recall$_{mac}$	-	**0.572**	**0.573**	-	**0.577**	**0.574**

5.2.6 Impact of Dataset Imbalance. Many previous datasets suffer from heavy class imbalance [2, 33] while the CBD is relatively balanced. To evaluate how dataset imbalance impacts our models, we experimented with the imbalance ratio of our training set. We split CBD into train (4,636 NCS, 1,893 CFS), validation (546 NCS, 180 CFS), and test (1,728 NCS, 691 CFS), then we subsampled the CFS or NCS samples in our training set to achieve a given ratio. Using the subsampled training sets, we trained our model using stratified k-fold cross validation ($k = 4$) and evaluated on the test set using the best model checkpoint for each fold. We experimented with the following CFS imbalance ratios: 1%, 5%, 10%, 29% (full CBD), 50%, 70%, 90%, 95%, 99%. These results are reported in Table 12.

Table 12. **Dataset imbalance study.** CB-BB and CB-BBA perform comparably on highly imbalanced datasets, where either CFS or NCS is overrepresented. Note that **red** and **green** numbers indicate row-wise minima and maxima, respectively.

CFS Ratio (%)	CFS Samples	NCS Samples	BB $F1_{mac}$	BBA $F1_{mac}$
1%	4636	48	**0.678**	**0.697**
5%	4636	245	**0.842**	**0.828**
10%	4636	516	**0.871**	**0.864**
29%	4636	1893	**0.878**	**0.887**
50%	1893	1893	**0.874**	**0.879**
70%	812	1893	**0.860**	**0.853**
90%	211	1893	**0.746**	**0.757**
95%	100	1893	**0.674**	**0.674**
99%	20	1893	**0.426**	**0.344**

5.2.7 Speed Benchmarks. Our speed benchmarks were obtained by running inference on 10,000 randomly-sampled sentences using the hardware identified in Section 4.6. Due to CB-DBA's high parameter efficiency (DistilBERT is only 6 layers deep), it *doubles* classification inference speed v.s. both CB-RBA and CB-BBA (column "claims/second" in Table 6) while outperforming CB-RBA in F1 score and performing on-par with CB-BBA in F1 score.

5.2.8 Performance Summary. To balance performance and speed, we select CB-DBA for production. Although CB-BBA has a slight F1 edge, CB-DBA is competitive and twice as fast.

5.3 Check-worthiness Score (*CWS*) Distribution Analysis

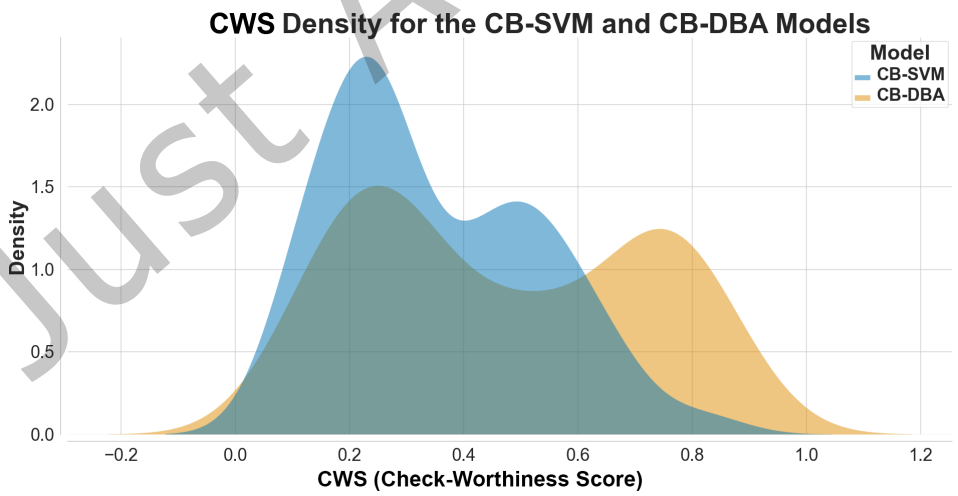

Fig. 7. Check-worthiness score density distribution on CB-DBA vs. CB-SVM using 100 Sentences from the January 14th, 2020 U.S. Democratic Presidential Debate

We also wanted to assess how our latest model compared with our previous production deployed model. The motivation behind this was to understand how the two models scored sentences with respect to each other.

To gain this understanding we analyzed the distribution of our models' outputs on a typical corpus of text, we processed 100 sentences from the January 14th, 2020 U.S. Democratic presidential debate. [15] Equal numbers of check-worthy and non-check-worthy sentences were chosen, such that we could check for two equally-sized peaks in the output distribution. Figure 7 displays the results, which use Kernel Density Estimation [38] to estimate the score distribution from discrete data points. Although both distributions are somewhat bimodal, the separation between the two peaks is more clearly differentiated in CB-DBA. This enables our users to more easily define a score threshold that suits their needs. Also note that the CB-SVM model's distribution clearly favors the NCS class despite being run on a dataset that is evenly constructed. From this, we can conclude that the CB-DBA model has improved CFS recall.

6 EVALUATION AGAINST REAL FACT-CHECKS

To evaluate the performance of ClaimBuster on real-life events, we compared CB-DBA's top ranked claims to those chosen by actual fact-checkers during 10 events of the 2020 U.S. Presidential Election cycle: 4 Democratic National Convention events, 4 Republican National Convention events, the Vice-Presidential Debate, and the first Presidential Debate. We first acquired the transcripts of these televised events from rev.com and split them up into sentences. Then, we identified sentences that were fact-checked by PolitiFact, The New York Times, or The Washington Post. In total, we obtained 13,044 sentences, of which 194 were fact-checked by PolitiFact, 248 by The Washington Post, and 378 by The New York Times.

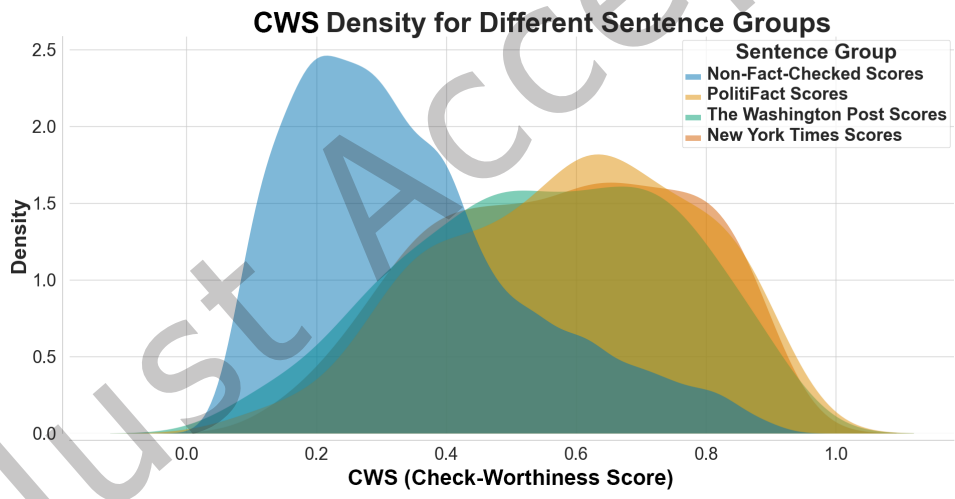

Fig. 8. CB-DBA check-worthiness score distributions for sentences that were and were not fact-checked by professional fact-checking outlets.

From Figure 8 we see that the score distributions, again produced via KDE, for the different organizations are very similar. This shows that, from the perspective of the model, organizations are consistent in how they choose claims to fact-check. By contrast, the score distribution for sentences that were not chosen for fact-checking is clustered to the left. This tells us that our model captures an understanding of "check-worthiness" which

[15] https://bit.ly/3bH4fL9

aligns with that of professional fact-checking organizations. Moreover, Figure 8 provides valuable insight into choosing a check-worthiness threshold based on our use case. For example, a user wanting to capture most of the check-worthy statements might choose a threshold of 0.4, whereas one wanting to be more selective might choose 0.6, which would still capture roughly half of the check-worthy claims.

7 RELATED WORK

Many recent projects aim to build robust claim-spotting frameworks. Gencheva et al. [13] extended the feature set of ClaimBuster's SVM model [17].

Another follow-up study [21] built an online system called ClaimRank [16] that also prioritized sentences for fact-checking based on check-worthiness scores. Later, TATHYA [36] developed a multi-modal approach, in which multiple SVMs were trained on different clusters of a dataset, and the most confident SVM's output was chosen as the final prediction. Konstantinovskiy et al. [26] created a *new* benchmark claim-spotting dataset and trained a logistic regression model using InferSent [10] sentence embeddings. Fullfact is developing a system called Hawk [4], for which not many details have been released. Unfortunately, none of the aforementioned models provided code or reproducibility details. Therefore, we did not attempt to re-implement any works for direct comparison.

Apart from claim-spotting, there are several projects that are testing *end-to-end* automated fact-checkers, including ClaimBuster and ClaimPortal, as explained in Section 2. Squash [17] fact-checks live debates by converting speech to text and querying a database of pre-existing fact-checks. Fakta [18] checks claims against reliable web sources [32]. FullFact [19] clusters groups of similar claims.

There also exist various projects that use computing to aid fact-checkers in *disseminating* fact-checks to the general public, including Schema.org's ClaimReview [20] which organizes fact-checks into a unified database, FactStream [21] which compiles fact-checks into a smartphone application, and Fatima, [22] a bot built by Aos Fatos, a Brazilian fact-checking organization, that scans Twitter for tweets containing already-debunked misinformation and refers readers to relevant fact-checks.

8 CONCLUSION

We have presented our novel approach to detecting check-worthy factual claims using adversarially-trained transformer networks. In addition to achieving state-of-the-art results on challenging benchmarks, we demonstrate the real-world impact that ClaimBuster continues to have on computational journalism. This work has also resulted in the public release of our revamped datasets and models.

In the future, we are interested in exploring adversarial training as a *defense against malicious adversaries*. As a publicly deployed API, ClaimBuster may be susceptible to exploitation without incorporating mechanisms that improve its robustness. For example, it has been shown by [23] that a model's classification can be strongly influenced when certain words are replaced by their synonyms.

9 OPEN-SOURCE CODEBASE, DATASET, API, AND GOOGLE COLAB NOTEBOOKS

We publicly release several artifacts of the ClaimBuster project to facilitate reproducibility, educative outreach, and community engagement:

[16] https://claimrank.qcri.org/
[17] https://bit.ly/31YTfnJ
[18] https://fakta.app/
[19] https://fullfact.org/automated
[20] https://schema.org/ClaimReview
[21] https://www.factstream.co/
[22] https://fatima.aosfatos.org/

- Codebase (https://github.com/idirlab/claimspotter) of the claim-spotting models discussed in Section 4.3, with detailed instructions on training and running the models.
- Dataset (https://zenodo.org/record/3836810) for training and evaluating the models (i.e., the CBD dataset in Section 4.1), a collection of sentences labelled manually in-house by high-quality coders.
- Crowdsourcing platform (https://idir.uta.edu/classifyfact_survey/) for curating the above dataset.
- ClaimBuster API (https://idir.uta.edu/claimbuster/api/) with its documentation at https://idir.uta.edu/claimbuster/api/docs/. The end points `/api/v2/score/text/<input_text>` and `/api/v2/score/paragraphs/<input_text>` on the API are regularly updated to host the best-performing claim-spotting model to score arbitrary text, with and without individual sentences being segmented and scored separately.
- Google Colabs (https://bit.ly/2Na0llQ) for the claim-spotting models in Section 4.3 which allow users to interactively explore the models.
- ClaimBuster project website (https://claimbuster.org) with links to the aforementioned artifacts and the following demonstration systems and related projects that use ClaimBuster.
- Claim-spotting on U.S. Presidential Debates using ClaimBuster (https://idir.uta.edu/claimbuster/debates).
- End-to-end fact-checking (https://idir.uta.edu/claimbuster/factchecker/) which uses ClaimBuster to highlight checkworthy factual claims.
- ClaimPortal [27] (https://idir.uta.edu/claimportal/) which uses ClaimBuster for claim-spotting on tweets.

10 ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Grants 1719054, 1937143, and 2346261. We also extend our gratitude to the Texas Advanced Computing Center (TACC) for providing compute resources used in this work's experimentation.

REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, et al. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI*. 265–283.

[2] Tariq Alhindi, Brennan McManus, and Smaranda Muresan. 2021. What to Fact-Check: Guiding Check-Worthy Information Detection in News Articles through Argumentative Discourse Structure. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Haizhou Li, Gina-Anne Levow, Zhou Yu, Chitralekha Gupta, Berrak Sisman, Siqi Cai, David Vandyke, Nina Dethlefs, Yan Wu, and Junyi Jessy Li (Eds.). Association for Computational Linguistics, Singapore and Online, 380–391.

[3] Hunt Allcott and Matthew Gentzkow. 2017. *Social Media and Fake News in the 2016 Election.* Working Paper 23089. National Bureau of Economic Research.

[4] Mevan Babakar and Will Moy. 2016. *White paper: The State of Automated Factchecking.* Technical Report. FullFact. 36 pages. https://fullfact.org/media/uploads/full_fact-the_state_of_automated_factchecking_aug_2016.pdf

[5] Alberto Barrón-Cedeño, Firoj Alam, Andrea Galassi, Giovanni Da San Martino, Preslav Nakov, Tamer Elsayed, Dilshod Azizov, Tommaso Caselli, Gullal S. Cheema, Fatima Haouari, Maram Hasanain, Mucahid Kutlu, Chengkai Li, Federico Ruggeri, Julia Maria Struß, and Wajdi Zaghouani. 2023. Overview of the CLEF–2023 CheckThat! Lab on Checkworthiness, Subjectivity, Political Bias, Factuality, and Authority of News Articles and Their Source. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, Avi Arampatzis, Evangelos Kanoulas, Theodora Tsikrika, Stefanos Vrochidis, Anastasia Giachanou, Dan Li, Mohammad Aliannejadi, Michalis Vlachos, Guglielmo Faggioli, and Nicola Ferro (Eds.). Springer Nature Switzerland, Cham, 251–275.

[6] Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Nikolay Babulkov, Bayan Hamdan, Alex Nikolov, Shaden Shaar, and Zien Sheikh Ali. 2020. Overview of CheckThat! 2020 — Automatic Identification and Verification of Claims in Social Media. In *Proceedings of the 11th International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction (CLEF '2020).* Thessaloniki, Greece.

[7] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack

Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[9] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*. 402–408.

[10] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. arXiv:1705.02364

[11] Ciprian-Gabriel Cusmuliuc, Lucia-Georgiana Coca, and Adrian Iftene. 2020. UAICS at CheckThat! 2020: Fact-checking claim prioritization. (2020).

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[13] Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *RANLP*. 267–276.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv:1412.6572

[16] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. In *SIGKDD*. 1803–1812.

[17] Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *CIKM*. 1835–1838.

[18] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, and et al. 2017. ClaimBuster: The First-ever End-to-end Fact-checking System. *PVLDB* 10, 12 (Aug. 2017), 1945–1948.

[19] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. arXiv:cs.CL/2111.09543

[20] Michael Heilman. 2011. *Automatic Factual Question Generation from Text*. Ph.D. Dissertation. USA. Advisor(s) Smith, Noah A.

[21] Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting Check-Worthy Claims in Arabic and English. In *NAACL*. 26–30.

[22] Damian Jimenez and Chengkai Li. 2018. An Empirical Study on Identifying Sentences with Salient Factual Statements. In *IJCNN*. 1–8.

[23] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932* (2019).

[24] Anna Kata. 2011. Anti-vaccine activists, Web 2.0, and the postmodern paradigm - An overview of tactics and tropes used online by the anti-vaccination movement. *Vaccine* 30 (12 2011), 3778–89.

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[26] Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. arXiv:1809.08193

[27] Sarthak Majithia, Fatma Arslan, Sumeet Lubal, Damian Jimenez, Priyank Arora, Josue Caraballo, and Chengkai Li. 2019. ClaimPortal: Integrated Monitoring, Searching, Checking, and Analytics of Factual Claims on Twitter. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 153–158.

[28] J Martinez-Rico, Lourdes Araujo, and Juan Martinez-Romo. 2020. NLP&IR@ UNED at CheckThat! 2020: A preliminary approach for check-worthiness and claim retrieval tasks using neural networks and graphs. (2020).

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[30] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. arXiv:1605.07725

[31] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[32] Moin Nadeem, Wei Fang, Brian Xu, Mitra Mohtarami, and James Glass. 2019. FAKTA: An Automatic End-to-End Fact Checking System. In *NAACL*.

[33] Preslav Nakov, Giovanni Da San Martino, Tamer Elsayed, Alberto Barrón-Cedeño, Rubén Míguez, Shaden Shaar, Firoj Alam, Fatima Haouari, Maram Hasanain, Watheq Mansour, Bayan Hamdan, Zien Sheikh Ali, Nikolay Babulkov, Alex Nikolov, Gautam Kishore Shahi, Julia Maria Struß, Thomas Mandl, Mucahid Kutlu, and Yavuz Selim Kartal. 2021. Overview of the CLEF–2021 CheckThat! Lab on Detecting Check-Worthy Claims, Previously Fact-Checked Claims, and Fake News.

[34] Brendan Nyhan and Jason Reifler. 2015. Estimating fact-checking's effects. (2015). https://www.americanpressinstitute.org/wp-content/uploads/2015/04/Estimating-Fact-Checkings-Effect.pdf

[35] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

[36] Ayush Patwari, Dan Goldwasser, and Saurabh Bagchi. 2017. TATHYA: A multi-classifier system for detecting check-worthy statements in political debates. In *CIKM*. 2259–2262.

[37] Gordon Pennycook and David G Rand. 2019. Fighting misinformation on social media using crowdsourced judgments of news source quality. *Proceedings of the National Academy of Sciences* 116, 7 (2019), 2521–2526.

[38] Murray Rosenblatt. 1956. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics* 27, 3 (09 1956), 832–837.

[39] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

[40] Marcin Sawiński, Krzysztof Węcel, Ewelina Paulina Księżniak, Milena Stróżyna, Włodzimierz Lewoniewski, Piotr Stolarski, and Witold Abramowicz. 2023. OpenFact at CheckThat! 2023: Head-to-Head GPT vs. BERT - A Comparative Study of Transformers Language Models for the Detection of Check-worthy Claims.

[41] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* 307 (2018), 195–204.

[42] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? arXiv:1905.05583

[43] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A Survey on Deep Transfer Learning. *Lecture Notes in Computer Science* (2018), 270–279.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762

[45] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45.

[46] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144

[47] John Zarocostas. 2020. How to fight an infodemic. *The lancet* 395, 10225 (2020), 676.