Rumour Detection in the Wild: A Browser Extension for Twitter

Rumour Detection in the Wild: A Browser Extension for Twitter

Andrej Jovanović and Björn Ross
University of Edinburgh
Edinburgh, United Kingdom
contact.me.maddox@gmail.com
b.ross@ed.ac.uk

Abstract

Rumour detection, particularly on social media, has gained popularity in recent years. The machine learning community has made significant contributions in investigating automatic methods to detect rumours on such platforms. However, these state-of-the-art (SoTA) models are often deployed by social media companies; ordinary end-users cannot leverage the solutions in the literature for their own rumour detection. To address this issue, we put forward a novel browser extension that allows these users to perform rumour detection on Twitter. Particularly, we leverage the performance from SoTA architectures, which has not been done previously. Initial results from a user study confirm that this browser extension provides benefit. Additionally, we examine the performance of our browser extension's rumour detection model in a simulated deployment environment. Our results show that additional infrastructure for the browser extension is required to ensure its usability when deployed as a live service for Twitter users at scale[1].

1 Introduction

The advent of social media has forever transformed the way in which we are able to communicate with one another. Content-sharing has effectively been democratised with the removal of existing traditional barriers (Bates, 2007). Indeed, this provides many benefits: for those individuals involved in a newsworthy event, for example, information can be shared in real-time. This allows for news to propagate faster, with the intention of informing the wider public and inciting action from relevant stakeholders. However, with the low barrier of entry to content production and dissemination on social media, the overall quality of information on these platforms has degraded (Shu et al., 2017).

[1]We make all the materials related to this work available at the following GitHub repository under an open-source license.

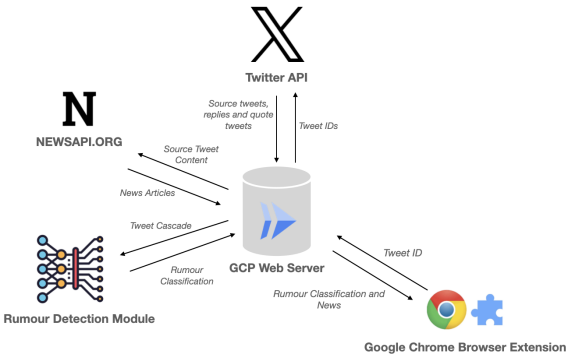


Figure 1: Server architecture of the Twitter rumour detection browser extension. Icons are taken from the following sources a, b, c, d and e

Users of social media platforms are now able to, inadvertently or deliberately, publish misinformation and falsehoods (Kydd and Shepherd, 2023). This has led to very damaging consequences for society in the past: Sharma et al. (2019) have provided examples of these effects in the financial, political and social domains. With this in mind, the machine learning community has created automatic detection systems which are designed to identify misinformation on social media platforms. However, it is the case that companies who own and manage these platforms are the ones that implement and deploy their own misinformation detection services (Kydd and Shepherd, 2023; Kumar et al., 2020). As such, regular content-consumers do not have access to these services directly. This takes away the autonomy of an individual who wishes to perform rumour detection for themselves, for example.

In this work, we present a Google Chrome browser extension, and the associated server architecture, (seen in Figure 1) that addresses this problem as highlighted by Fernandez and Alani (2018). It allows regular Twitter users to perform on-demand rumour detection for any tweet, whilst enhancing their experience through proving semantically-related news articles.

2 Related Work

Considerable progress has been made in investigating methods to perform rumour detection on social media platforms. Seminal papers in this discipline initially focussed on generating informative, handcrafted features for this task (Castillo et al., 2011; Qazvinian et al., 2011; Yang et al., 2012). Stemming from these works, researchers refocussed their attention on temporal features. Kwon et al. (2013) showed that these features are highly predictive of rumours as they described the periodic bursts that are typical for these phenomena. These new features now allowed researchers to capture how rumours change over time, which is particularly important for early rumour detection. Ma et al. (2015) identified rumours through modelling their lifecycle as a time series, whilst Wu et al. (2015) approached this through modelling the propagation pattern as a tree. Furthermore, certain features were shown to have greater importance at different steps in a rumours' propagation (Kwon et al., 2017). However, the collective flaw in these works is that the feature engineering processes are detail-specific, could introduce biases and are extremely laborious (Bian et al., 2020; Ma et al., 2016). As such, SoTA solutions turn to deep learning for the automatic feature representations, increased model complexity and subsequent increase in performance for rumour detection.

Ma et al. (2016) implemented the same time series ideas from their earlier works (Ma et al., 2015) with recurrent neural networks (RNNs), later improved with attention (Chen et al., 2017), which leveraged the deep hidden representations that were learnt by the neural network. Ma et al. (2018) found that recursive neural networks (RvNNs) were more performant than RNNs as they are able to embed both content-based and propagation-based information due to their tree-like structure. This architecture was also improved with an attention mechanism (Ma et al., 2020). Bian et al. (2020) achieved SoTA performance on the *Twitter15* and *Twitter16* datasets for rumour detection on Twitter through not only modelling the propagation properties of a rumour, but also the dispersion properties with their Bi-Directional Graph Convolutional Network (Bi-GCN).

However, deploying said rumour detection models as a service has not been extensively researched in an academic setting. Gupta et al. (2014) introduced TweetCred, a Google Chrome browser extension that assigned a credibility score to each tweet on a user's feed using an SVM-rank model using 45 handcrafted features. Thilakarathna et al. (2020) created a browser extension for Twitter, called Veritas, that is able to detect fake news on the social media platform. Most notably, their architecture involves a model trainer pipeline to ensure that their neural models are constantly up-to-date. Kydd and Shepherd (2023) explored a very similar tool that used a deep learning solution as the backbone to a browser extension focusing on clickbait detection.

Additionally, we note that news articles are a typical resource that one would use to determine the status of a rumour. These resources enhance the experience for the user, informing them of the context in which the tweet occurs. To this end, our work attempts to address the following flaws found in previous work. i) Previous works that tackle rumour detection specifically do not leverage SoTA rumour detection models at the core of their service, which renders them outdated. ii) In certain cases, the rumour detection model is deployed on the user-side. We posit that with SoTA models, this will render the service inutile due to the computational complexity of the models (Kydd and Shepherd, 2023). iii) None of the aforementioned works enhance the user's experience through recommending articles that are semantically related to the tweet in question. Our browser extension will explore this addition.

3 Browser Extension Architecture

When creating a browser extension, and the associated architecture, we wish for the following properties to be met. These are chosen to maximise the extension's ease-of-use, performance and informativeness.

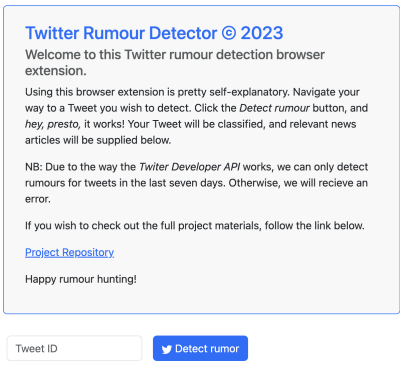
D1: Twitter users should be able to use the service in-real time in concert with their ordinary browsing experience.

D2: Users of the service should not have to bear the computational load of the underlying architecture.

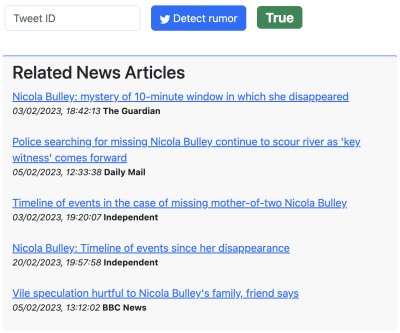
D3: The detection model should be interchangeable, allowing for the browser extension to improve continually with the latest advancements in the field.

D4: News articles that are semantically related to the source tweet should enhance the user's rumour detection experience.

To meet the above desiderata, we put forward the



(a) The browser extension's landing page.



(b) The browser extension with results: a rumour classification label of "True" and a list of five semantically related news articles.

Figure 2: Screenshots showing the graphical user interface (GUI) of the browser extension.

browser extension architecture seen in Figure 1. Furthermore, we designed a user-friendly interface in the form of a Google Chrome browser extension seen in Figure 2.

A user will interact with the system as follows:

(i) A user browses Twitter via Google Chrome. Once they identify a particular tweet on which they wish to perform rumour detection, the user opens the browser extension and clicks the "Detect rumour" button seen in Figure 2a (**D1**). Once the button has been pressed, the tweet ID is extracted from the URL of the tweet and the client's web browser sends a POST request to the web server (**D2**), with the specific tweet ID as a parameter.

(ii) Once the web server receives this POST request, three functions occur sequentially:

  (a) Using the tweet ID, the web server interacts with the Twitter API to retrieve the source tweet, and its respective tweet cascade as outlined in section 3.2.

  (b) Once the tweet cascade has been retrieved and preprocessed, this is then fed into the rumour detection model (**D3**) seen in section 3.1, and inference is performed. This returns a particular rumour classification label.

  (c) Finally, using the source tweet, the web server finds semantically-related keywords and retrieves relevant news articles from NewsAPI.org (**D4**), as outlined in Section 3.3. This returns a list of relevant news articles to the web server.

(iii) Once the rumour classification label and the relevant news articles have been retrieved, these are returned to the user as a JSON object in response to the original POST request.

3.1 Rumour Detection Model Choice

For this iteration of the rumour detection browser extension, we choose the Bi-GCN architecture (Bian et al., 2020) as the machine learning model used in our service. The training code can be found in our GitHub repository, or at in the original repository. This model was chosen as it reported SoTA performance for the rumour detection task on the *Twitter15* and *Twitter16* datasets. The model is able to represent both the top-down and bottom-up views of a tweet cascade, each of which is passed to a dedicated two-layer GCN, along with a shared tweet feature matrix (linguistic tokens). Bian et al. (2020) also implement DropEdge to prevent overfitting, and root feature enhancement after each GCN layer to emphasise the information contained in the source tweet[2]. To deploy the model, we train it using the *Twitter16* data, with the same specifications as in (Bian et al., 2020) (see Section 4) with 5-fold cross validation, taking the average model as final. The Bi-GCN, with a hidden dimension size of 64, is trained using stochastic gradient descent with the Adam optimiser ($\eta = 5 \times 10^{-4}$) to minimise the cross entropy loss. The model is trained for 200 epochs, with early stopping on the validation loss and patience set to 10 epochs. DropEdge rate is set to 0.2, dropout rate is set to 0.5 and L2 regularisation is applied to all model parameters with $\lambda = 1 \times 10^{-4}$.

[2]We point the interested reader to their original paper for more details on the model's functionality.

3.2 Twitter API

We make use of the Twitter API to retrieve the raw data required to classify the rumour status of a particular tweet. Since the Bi-GCN represents each tweet as a cascade, we first collect the source tweet from the API to act as the root of the cascade. We then retrieve all the replies, quote tweets and retweets related to the root. We continue this process recursively, until the algorithm bottoms out at the leaf tweets. Once we have retrieved the cascade, each tweet is assigned its textual features according to the vocabulary used at training (Section 4).

3.3 Semantically-Related News Articles

To find the semantically-related news articles, we made use of the open-source KeyBERT tool[3]. This package leverages embeddings that are created using a Sentence-BERT architecture (Reimers and Gurevych, 2019; Devlin et al., 2018), particularly the *all-MiniLM-L6-v2* model found on HuggingFace[4], to generate the semantically related keywords. Before passing raw tweet text to KeyBERT, we first preprocess it with NLTK's tweet tokenizer[5]. Candidate keyword phrases are extracted from N-gram sequences (one to three grams in particular) in the document text, and word embeddings are computed for these. KeyBERT returns the candidate phrases that are most similar to the document text using the cosine similarity metric, and have been reranked using Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to increase the diversity. The keywords are then passed to NewsAPI.org which returns relevant news articles according to a keyword-based query.

4 Datasets

We make use of the *Twitter15* and *Twitter16* datasets (Ma et al., 2017) to train our rumour detection model. The datasets comprise rumours linked to newsworthy events at specific time periods; the statistics of these can be found in Table 1. In particular, these datasets are graphical in nature. Each node refers to a tweet, where each node is described by textual features derived from a pretrained vocabulary of the top 5000 words in terms of TF-IDF (Sammut and Webb, 2010) score. Edges between tweets represent their retweet or response relationships. A collection of tweets in a cascade describes an event, and is assigned a veracity tag (rumour **(UR)**, non-rumour **(NR)**, false rumour **(FR)** or true rumour **(TR)**) which were derived from cross-referencing rumour debunking websites.

Statistic	Twitter15	Twitter16
# of posts	331,612	204,820
# of users	276,663	173,487
# of cascades	1,490	818
# of non-rumours	374	205
# of false rumours	370	205
# of true rumours	372	205
# of unverified rumours	374	203
Avg. time length / cascade (Hours)	1,337	848
Avg. # of posts / cascade	223	251
Max # of posts / cascade	1,768	2,765
Min # of posts / cascade	55	81

Table 1: Statistics of the *Twitter16* and *Twitter16* datasets.

5 Rumour Detection Model Evaluation

5.1 Out-of-Distribution Performance

An imperative part of the browser extension's functionality relies on the underlying performance of the machine learning model used for inference. It is well accepted that models are able to generalise well (if trained appropriately) to data that is unseen, but comes from a similar distribution to the training data (Hendrycks and Dietterich, 2019; Klaise et al., 2020; Engstrom et al., 2019). However, using our browser extension for rumour detection in the wild necessitates that the model will be used to evaluate tweets that are OOD relative to the data which it was trained. Concept drift is frequently occurring in social media, particularly when the nature of discourse underlying different rumours changes (Horne et al., 2019). Furthermore, end-users of the browser extension could use the tool outside the environment in which it was intended to be deployed. In these scenarios, we would expect that the browser extension would perform suboptimally on the rumour detection task.

To simulate this effect, we conduct a data mixing experiment. Specifically, we leverage two datasets that are frequently occurring in the rumour detection literature: *Twitter15* and *Twitter16*. In this experiment, we create a third *TwitterMix* dataset through a linear-interpolation-like combination of the *Twitter15* and *Twitter16* datasets $TM = p_{T15} * T_{15} + p_{T16} * T_{16}$, controlling for

[3] KeyBERT
[4] all-MiniLM-L6-v2
[5] NLTK Tokenize

the size of the dataset by enforcing the following constraint: $p_{T15} + p_{T16} = 1$. p_{T15} and p_{T16} act as the proportion of the dataset that is selected. We posit that a model trained on *TwitterMix*, as a simple baseline, would be able to mitigate partially the effects of concept drift. This baseline is akin to a single-shot retraining procedure. Following the training regime set out in Section 3.1, we train three separate models based on the following datasets: i) a *Twitter15* model on the unmixed $1 - p_{T15}$ *Twitter15* data, ii) a *Twitter16* model on the unmixed $1 - p_{T16}$ *Twitter16* data, and iii) a *TwitterMix* model on the mixed data. In the case of the *Twitter15* and *Twitter16*, all models were evaluated on the p_{T15} and p_{T16} data. All models were evaluated on the *TwitterMix* data, where we report the cross-validation performance in the case of the *TwitterMix* model. In Figure 3, we view the experimental results where we set $p_{T15} = p_{T16} = 0.5$. This particular proportion was chosen for the sake of simplicity. On both the *Twitter15* and *Twitter16* datasets, we find that their respective models perform well, as expected. Similarly, we find that the *Twitter15* model's performance decays dramatically on the *Twitter16* data, and *vice versa*. Both models perform equally well on the *TwitterMix* data. Furthermore, we find that the *TwitterMix* model is able to mitigate some adverse effects when evaluating a model on OOD data, seen particularly on the *Twitter15* data. While not as performant as the *Twitter16* model on *Twitter16* data, it is able to recover some performance relative to the *Twitter15* model. These results confirm findings of Horne et al. (2019); Paleyes et al. (2022); Lobo et al. (2020), and show the importance of retraining schedules and engines when deploying machine learning models in browser extension tools (Thilakarathna et al., 2020). We perform additional experiments (seen in Appendix B) altering the proportion of p_{T15} and p_{T16}. These experiments show that if adequate care is not placed on constantly maintaining a representative/diverse training sample through retraining, or if additional methods are not put into place to detect when samples are OOD, the performance of the model decays significantly.

5.2 Imperfect Data Performance

5.2.1 Textual Ablation Experiments

Another implication to consider is that the rumour detection model's performance is also constrained by the Twitter API limits. For example, the quote tweet endpoint has a 75 request per 15-minute window threshold[6]. This would not affect those users who wish to perform rumour detection on a handful of tweets, but rather "power users". In the cases, we could observe that the true tweet cascade cannot be sufficiently represented, which in turn could affect the performance of the rumour detection model. Similarly, due to the effects of concept drift mentioned in Section 5.1, the rumour detection model would be unable to represent the textual content of certain tweets in a rumour cascade. As seen in Section 3.1, the Bi-GCN architecture is trained on a fixed, static vocabulary (as are many other NLP solutions). If a tweet were composed of tokens that were out-of-vocabulary for the rumour detection model, this tweet would then be underrepresented.

To simulate these effects, we run two ablation studies. First, we run a text ablation study where generate new versions of the *Twitter15* and *Twitter16* datasets according to some textual ablation proportion. In these new datasets, we randomly replace, according to the specified proportion, the textual features of a given tweet with `[0:1]`, which is the corresponding `[index:count]` pair for an `<END>` tag. Once the new datasets have been created, we train a *Twitter15* and *Twitter16* model, and report the performance on five-fold cross validation as done in section 3.1. This was done for the textual ablation proportions: 0%, 50%, 70%, 90%, 100%. We view the results of this experiment in Figure 4.

We see that the performance of the two models dramatically decays as we increase the proportion of tweets that have their textual features removed. These results were expected due to the importance of textual features to rumour detection models. Textual features have always provided an important signal in predicting the rumour status of a particular tweet (Section 2). The same is true with the Bi-GCN model. These results stress that without the proper, and full, representation of tweet cascades, the performance of the rumour detection model drops dramatically.

5.2.2 Node Ablation Experiment

Similar to the textual feature ablation experiment, we conduct a node ablation experiment. We first generate new versions of the *Twitter15* and *Twitter16* datasets according to some node ablation proportion. However, instead of removing textual

[6] Quote Tweets API

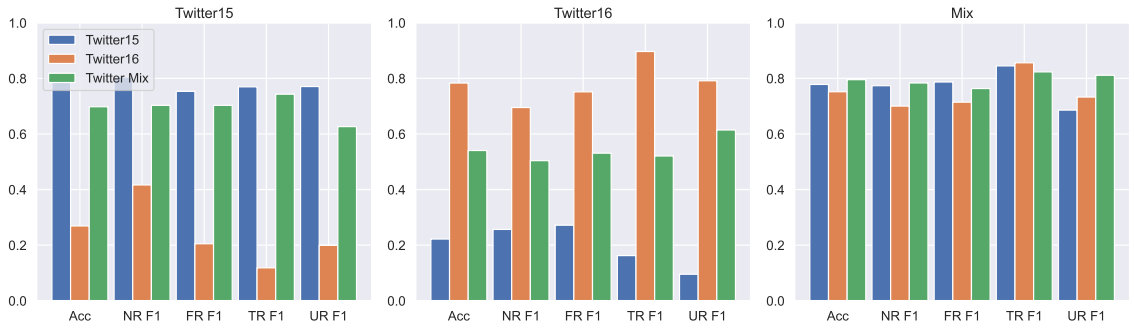


Figure 3: Result from the dataset mixing experiment for the *Twitter15*, *Twitter16* and TwitterMix models, with $p_{T15} = p_{T16} = 0.5$. Each subplot indicates the evaluation dataset, and the legend the model versions.

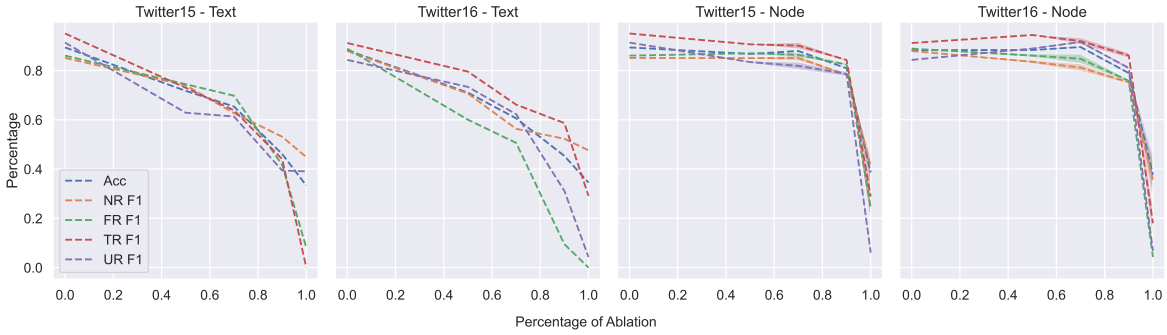


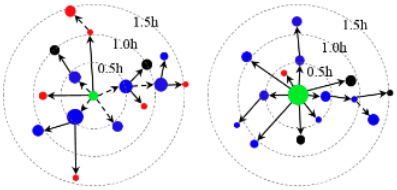
Figure 4: Results from the textual and node ablation experiments, across varying ablation proportions, for the *Twitter15* and *Twitter16* datasets.

features, we randomly remove nodes, and their descendents, from every tweet cascade[7]. This was done to simulate the scenario where certain tweets would not be retrieved by the Twitter API. We train a *Twitter15* and *Twitter16* model on these new datasets and report the performance on five-fold cross validation as done in section 3.1. We repeat this experiment for the following node ablation proportions: 0%, 50%, 70%, 90% and 99.9%. The results of these experiments are specified in Figure 4.

Surprisingly, we did not observe the same effect as was seen in the textual ablation experiments (Figure 4). Instead, we see that the Bi-GCN was able to was able to maintain, and sometimes improve, performance relative to the baseline, across both datasets. This was achieved until over 90% of the tweets in each cascade had been removed. After this point, the models' predictive capability sharply decreased – once enough tweets had been removed, both models lose enough signal from the input data to classify the rumour status accurately. However, we can contrast these results with the early rumour detection study in the original paper by Bian et al. (2020). Their work examined the Bi-GCN's performance on the task of early rumour detection. Although our experiment does not remove tweets from the cascade temporally (as we do not have access to the timestamp for each tweet), we are, essentially, creating an experiment that is very similar to this experiment in Bian et al. (2020).

However, an interesting result is that in both datasets, we observed that the models were able to score better than the baseline even with fewer tweets representing the cascade. When randomly removing a tweet, and its descendents, from a cascade, we have no rules enforcing what type of tweets are removed from the cascade. If we observe the rumour tweet in Figure 5, tweets that express doubt in response to a root tweet indicates that the tweet is potentially a rumour (Kwon et al., 2017). As such, removing the tweets that express support make the tweet seem more rumour-like. Similarly, removing the tweets that express doubt from the non-rumour would make this tweet more non-rumour like. These situations would make each of the tweet cascades seem more like a prototypical example of their respective class.

[7]We did not remove the root nodes from the tweet cascades as the label for each tweet cascade is tied to the root node.



(a) A rumor (b) A non-rumor

Figure 5: Prorogation structure of two source tweets taken from Ma et al. (2017). Red nodes express doubt, blue nodes express neutrality and black nodes express support. The green node is the root of the cascade.

A similar effect could, by chance, be observed in our node ablation experiments. By randomly removing certain nodes, we inadvertently simplify the rumour detection task for that tweet as it would seem more prototypical of its class.

6 Latency Experiments

Similar to the work done by Gupta et al. (2014), we wish to evaluate the browser extension's performance with respect to response time. This is calculated as the amount of time taken for our browser extension to respond to a particular rumour detection request. In our experiment, we measure the response time across 50 randomly selected newsworthy tweets of varying size. We view the cumulative distribution function (CDF) of the response times in Figure 6.

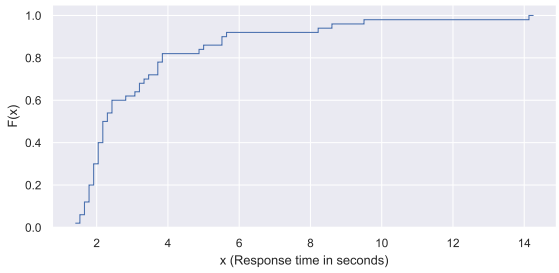


Figure 6: CDF for response time taken for rumour detection across 50 tweets.

Observing Figure 6, we see that our browser extension is able to provide a response for 90% of requests in six seconds or less. This is an improvement over the 82% of requests in Gupta et al. (2014); however, their experiment analysed the CDF across 5.4 million requests. Due to time constraints, our sample size is significantly smaller. Yet, we can make some initial comparisons between the two solutions. Our work retrieves the entire tweet cascade for a tweet, and predicts the rumour status using the Bi-GCN model. TweetCred, on the other hand calculates 45 handcrafted features based on the tweet itself, and uses SVM-rank to assign a credibility score. Given the fact that our approach uses far more information per tweet, and a more sophisticated and computationally intensive model[8], this is still an encouraging result for our browser extension. Unfortunately, the other browser extensions mentioned in Section 2 do not report results for a response time experiment. As such, we are unable to compare our extension to other solutions on this particular axis.

7 User Study

To determine whether the browser extension provides benefit to Twitter users, we ran an anonymised user study. We asked 19 participants to perform five rumour detection tasks (RDTs). In this context, a RDT is divided into two questions.

(i) Before performing rumour detection using the browser extension, determine the rumour status of the current tweet.

(ii) After performing rumour detection, assess whether the browser extension aid in determining the true status of the tweet.

Each set of questions was asked on a preselected tweet that was newsworthy at that instance in time. This was done to ensure that the tweets used in the study were as similar as possible to the training data distribution on which the model was trained (see Section 5.1 for out-of-distribution performance).

After performing these tasks, we asked the users to comment on their overall experience using the browser extension. The user study was facilitated through the use of anonymised survey. The users accessed the server architecture through a remote Google Cloud Platform server (GCP), and the browser extension itself from a shareable Google Chrome Store link. The user study was approved by a research ethics board (see Section 8). The results from the RDT and overall experience feedback are seen in Figures 7 and 8, respectively. See Appendix A for full details on the user study.

Prior to performing rumour detection, we see that there is considerable disagreement amongst the annotators. Particularly, we find that Randolph's Kappa is $\kappa = 0.345$ (Randolph, 2010). These results confirm an assumed truth in the field: rumour

[8]Most notably, our browser extension was able to classify a tweet cascade with over 579 retweets, 343 quote tweets and 454 replies in approximately 14 seconds.

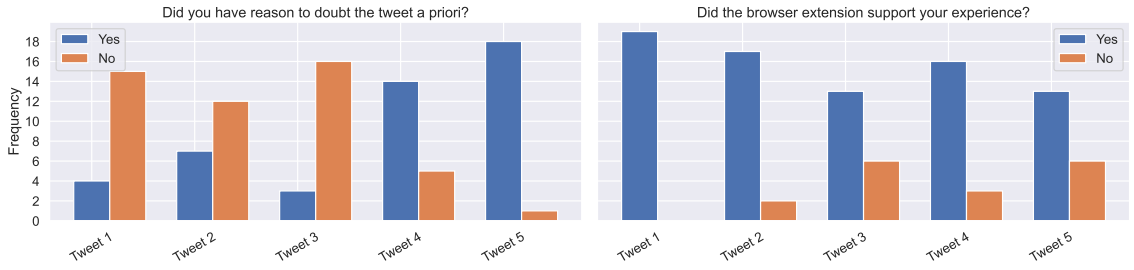


Figure 7: Results from the RDT in the user study.

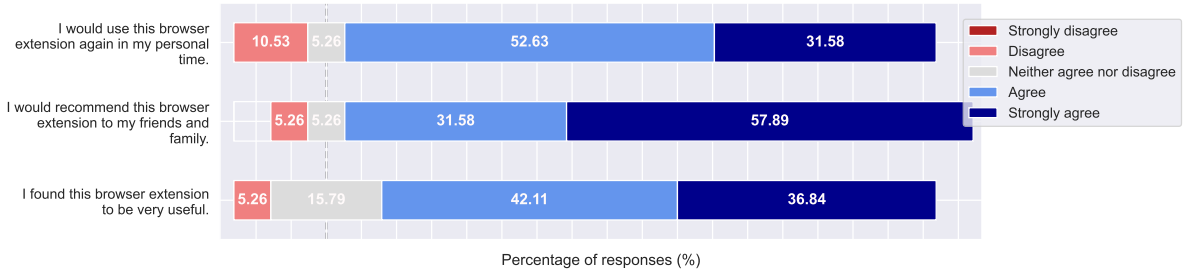


Figure 8: Results from global feedback in the user study.

detection, a highly subjective task, is difficult for humans as the labels are not necessarily objective (Touvron et al., 2023). This further motivates the need for assistive tools such as this browser extension. Contrastingly, we see that, generally, annotators found that the browser extension supported their rumour detection experience. $\kappa = 0.443$ which supports the fair agreement on the browser extension's positive impact on their rumour detection experience. An interesting result is that this kappa score is similar to the user agreement on the credibility score (43%) obtained for TweetCred (Gupta et al., 2014). Whilst the questions posed to the users are different, these results show that there is some benefit to be gained through using additional rumour detection tools. However, there needs to be additional measures put into place to make users more confident in the tool's performance, which would lead to higher user agreement (see Section 8).

Furthermore, we asked the users to rate their agreement to three questions on a five-point Likert scale. Observing Figure 8, we see that the feedback for the browser extension is generally positive. 78.95% of the participants in the user study agreed or strongly agreed with finding the browser extension to be useful. However, we see that there was a small portion of users who either disagreed or were ambivalent to the three statements. This study did not require the participant to be a Twitter user. As such, the study could have attracted participants who: do not use Twitter frequently, do not use Twitter as a news source, or those users that do not have a Twitter account at all. These users would not see the need to have access to a tool such as this browser extension as they would have no use for it personally.

8 Conclusion and Future Work

In this work, we put forward a novel browser extension that allows Twitter users to perform rumour detection, leveraging the performance from SoTA models. Our work shows that this tool provides benefit to those Twitter users wanting autonomy over their rumour detection. However, we note that our work is merely the first iteration in a series of deployments. Future work could explore additional mechanisms to allow the browser extension to cope with OOD data. Online retraining (Horne et al., 2019) has been shown to be effective in minimising the effects of concept drift; this is similar to the trainer pipeline in Veritas (Thilakarathna et al., 2020). Furthermore, Diethe et al. (2019) show a more sophisticated paradigm with their continual learning approach. Additionally, we could extend the browser extension's functionality through allowing its users to submit examples of tweets they believe to be (non-)rumours (with evidence) to some community-moderated data store. This process could be used to create more up-to-date datasets for rumour detection.

Limitations

The work suffers from two main limitations, the first of which is the current system's reliance on the Twitter API, and its changing access requirements. At the time of writing, Twitter API users will no longer be able to make use of the GET API endpoints, which include the endpoints used to fetch the information needed for a tweet's cascade representation (Section 3.2), under the free tier. Instead, users will have to pay $100 per month[9]. As such, this limits the use-case of the browser extension that we have created. However, the flexible architecture that we have created allows the browser extension to be ported to a different context. Instead of focussing on Twitter, a similar use-case would be found with Sina Weibo, for example. The browser extension could focus on fake news detection, rather than on rumour detection. Each of these disciplines have their own state-of-the-art solutions in the literature, and exploring practical tools that would leverage their performance would be a worthwhile research direction.

A second limitation lies in the small sample size of the user study. Furthermore, the participants themselves could be biased in their evaluation of the browser extension because of their relation to the author; the browser extension was shared via a university mailing list. However, due to the anonymity of the study, we hope that the participants of the user study would be objective in their assessment of the browser extension. Nevertheless, we find that these initial results support and encourage the viability of a Twitter rumour detection extension. However, a potential direction for future work would be extending the browser extension to a wider, more diverse audience.

Ethics Statement

This project obtained approval from the Informatics Research Ethics committee at the University of Edinburgh.
Ethics application number: 2023/260884
Date when approval was obtained: 2023-01-30

Acknowledgements

We are grateful to the following people (in no particular order) for their helpful insight and corrections to the manuscript: Alessandro Palmarini, Maxime Labonne, and Simon Chi Lok U.

[9] https://developer.twitter.com/en/portal/products/basic

References

Benjamin Bates. 2007. Yochai benkler. the wealth of networks: How social production transforms markets and freedom. *Journal of Media Economics*, 20:161–165.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):549–556.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 675–684, New York, NY, USA. Association for Computing Machinery.

Tong Chen, Lin Wu, Xue Li, Jun Zhang, Hongzhi Yin, and Yang Wang. 2017. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *CoRR*, abs/1704.05973.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. 2019. Continual learning in practice.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2019. Exploring the landscape of spatial robustness. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1802–1811. PMLR.

Miriam Fernandez and Harith Alani. 2018. Online misinformation: Challenges and future directions. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 595–602, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, and Patrick Meier. 2014. Tweetcred: Real-time credibility assessment of content on twitter. pages 228–243.

Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Benjamin D. Horne, Jeppe Nørregaard, and Sibel Adali. 2019. Robust fake news detection over time and attack. *ACM Trans. Intell. Syst. Technol.*, 11(1).

Janis Klaise, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, and Alexandru Coca. 2020. Monitoring and explainability of models in production.

Sachin Kumar, Rohan Asthana, Shashwat Upadhyay, Nidhi Upreti, and Mohammad Akbar. 2020. Fake news detection using deep learning models: A novel approach. *Trans. Emerg. Telecommun. Technol.*, 31(2).

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PLOS ONE*, 12(1):1–19.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108.

Marc Kydd and Lynsay A. Shepherd. 2023. Deep breath: A machine learning browser extension to tackle online misinformation.

Jesus L. Lobo, Javier Del Ser, Albert Bifet, and Nikola Kasabov. 2020. Spiking neural networks and online learning: An overview and perspectives. *Neural Networks*, 121:88–100.

Jing Ma, Wei Gao, Shafiq Joty, and Kam-Fai Wong. 2020. An attention-based rumor detection model with tree-structured recursive neural networks. *ACM Trans. Intell. Syst. Technol.*, 11(4).

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Jim Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 1751–1754, New York, NY, USA. Association for Computing Machinery.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 708–717, Vancouver, Canada. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. 2022. Challenges in deploying machine learning: A survey of case studies. *ACM Comput. Surv.*, 55(6).

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Justus Randolph. 2010. Free-marginal multirater kappa (multirater κfree): An alternative to fleiss fixed-marginal multirater kappa. volume 4.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US, Boston, MA.

Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. 2019. Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3).

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

Madusha Prasanjith Thilakarathna, Vihanga Ashinsana Wijayasekara, Yasiru Gamage, Kavindi Hanshani Peiris, Chanuka Abeysinghe, Intizar Rafaideen, and Prathieshna Vekneswaran. 2020. Hybrid approach and architecture to detect fake news on twitter in real-time using neural networks. In *2020 5th International Conference on Information Technology Research (ICITR)*, pages 1–6.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ke Wu, Song Yang, and Kenny Q. Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12, New York, NY, USA. Association for Computing Machinery.

A User Study Information

The participants, all of whom are fluent in English, accessed the survey via an anonymised Google Form sent via a university undergraduate mailing list. The web server architecture was deployed on Google Cloud Platform; all participants received the same underlying tweet cascade and recommended articles as a response for every tweet. The users could not comment on the speed of the service as these results were cached for the sake of reproducibility and efficiency.

B OOD: Additional Proportion Experiments

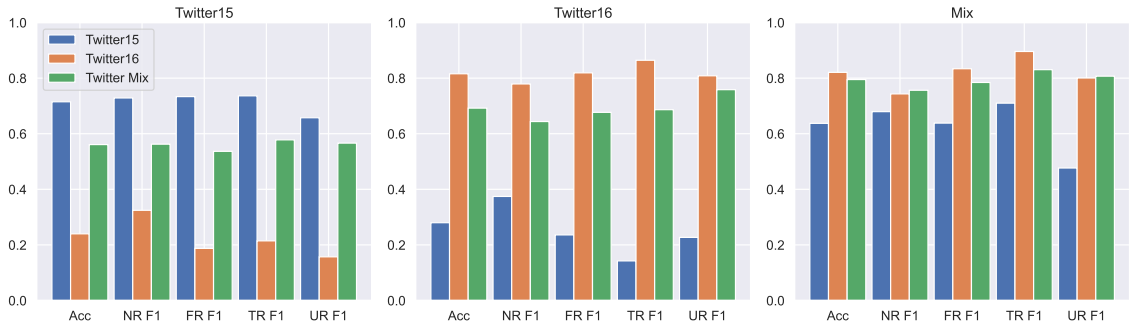


Figure 9: Result from the dataset mixing experiment for the *Twitter15* and *Twitter16* models, with $p_{T15} = 0.3$ and $p_{T16} = 0.7$.

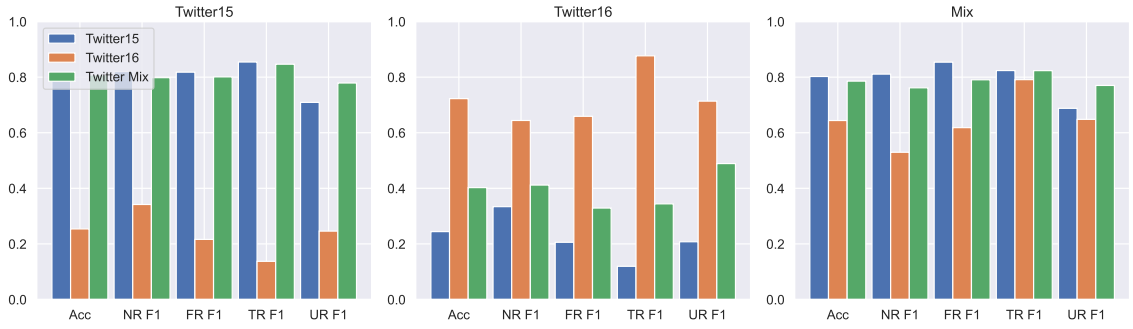


Figure 10: Result from the dataset mixing experiment for the *Twitter15* and *Twitter16* models, with $p_{T15} = 0.7$ and $p_{T16} = 0.3$.