Information Processing and Management 60 (2023) 103414

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

This is not new! Spotting previously-verified claims over Twitter

Watheq Mansour [a,*], Tamer Elsayed [a], Abdulaziz Al-Ali [a,b]

[a] Computer Science and Engineering Department, Qatar University, Qatar
[b] KINDI Center for Computing Research, Qatar University, Qatar

ARTICLE INFO

Dataset link: https://github.com/Watheq9/SpotVC

Keywords:
Verified claim checking
Fact checking
Fake news detection
Efficiency
Retrieval
Classification

ABSTRACT

Several fake claims are commonly repeated over time, especially on social media. To identify such previous claims, the verified claim retrieval task was studied, where, for a given input claim, the goal is to find previously-verified claims that are relevant to it. However, this view assumes that *each* claim was already verified, which may not be true for all claims in the real-world scenario. In this work, we introduce the Verified Claim Checking problem over Twitter, in which the relevant verified claims are retrieved *only if* the input claim was indeed previously-verified, thus saving computation time. We address the problem by proposing *SpotVC*, an end-to-end approach consisting of two stages, namely a filter and a reranker. The proposed filter achieved an average F_1 of 0.81 while significantly reducing computation time. Moreover, the proposed reranker outperformed the state-of-the-art models on two public datasets and provided on-par performance on a third one. Overall, our proposed system exhibits an effective operational balance in the trade-off between efficiency and effectiveness for the real-world scenario.

1. Introduction

In the last decade, the world has experienced a growing epidemic of misinformation. The large volume of fake news and its rapid dissemination over social networks posed severe threats, not only to government and organizations but also to individuals. For example, a man carried an AR-15 rifle in the Washington DC Pizzeria because of online news stating that a pizzeria was harboring young children as sex slaves (Kang & Goldman, 2016). Toward limiting the large-scale weaponization of misinformation, many fact-checking organizations have arisen in the last few years, e.g., FactCheck.org,[1] Snopes,[2] and others. However, performing fact-checking manually involves extensive labor and consumes a considerable amount of time. Moreover, the capacity of those organizations cannot keep up with the high rate at which fake news is spreading over online social networks. To address these limitations, researchers propose to build automated fact-checking systems that recognize check-worthy input and examine their veracity (Thorne & Vlachos, 2018; Vosoughi et al., 2018).

As a component of a claim verification pipeline, Shaar et al. (2020) proposed the verified claim retrieval problem. Fig. 1 shows an example of a tweet and its relevant verified claim. From the figure, we can notice that the task is not straightforward for the machine to capture the relevance, as the claim mentioned within the tweet can be expressed in different forms. Generally, a claim retrieval system aims to retrieve all previously-checked claims with respect to an input claim. The need for such systems is attributed

* Corresponding author.
E-mail addresses: wm1900793@qu.edu.qa (W. Mansour), telsayed@qu.edu.qa (T. Elsayed), a.alali@qu.edu.qa (A. Al-Ali).
URL: https://sites.google.com/view/bigir/members/watheq-mansour (W. Mansour).

[1] http://www.factcheck.org/.
[2] http://www.snopes.com/.

https://doi.org/10.1016/j.ipm.2023.103414
Received 5 February 2023; Received in revised form 16 May 2023; Accepted 21 May 2023
Available online 3 June 2023

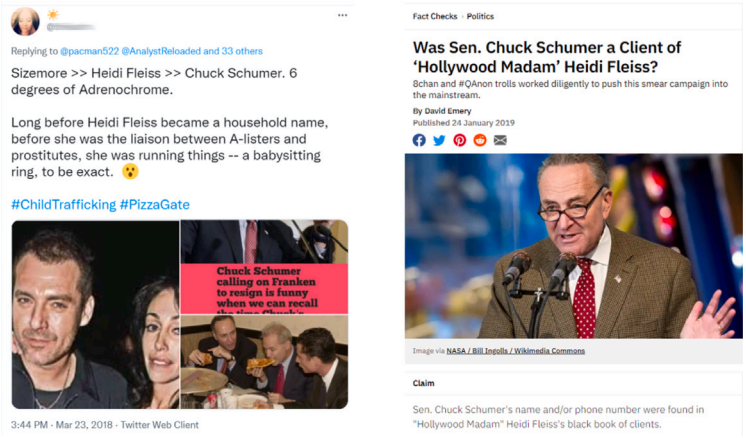


Fig. 1. A verified claim retrieval example: input (left) and relevant verified claim (right).

to the fact that many viral claims keep repeating at different times with variant surface forms. Therefore, identifying those claims could dramatically mitigate the negative impact of fake news. Additionally, it will allow more time to be spent on the new claims.

To that end, the verified claim retrieval problem was offered as a shared task in CheckThat! lab 2020 (Barrón-Cedeño et al., 2020), 2021 (Nakov et al., 2021), and 2022 (Nakov et al., 2022). Several teams participated and proposed different systems with reasonable effectiveness. However, research addressing this problem has always assumed that there is at least one relevant fact-checked claim for every single input claim. While this is a simplifying assumption, it does not reflect the real-world scenario, as some (or most) claims are new and were not previously verified. As such, an important question arises: What if the input claim was not previously checked? To the best of our knowledge, there is no prior work that addresses such a question. A system that detects whether the input claim has been verified before or not is indeed beneficial, since it serves as a filter that saves the time of running subsequent components in a fact-checking system pipeline and also the time of the user by not having to filter false positives manually.

In this paper, we introduce the *Verified Claim Checking* problem to match the actual real-world scenario. The proposed problem takes into consideration the case when there are no relevant verified claims for a given input. It then has two sub-problems, verified claim *detection* and verified claim *retrieval*. The *detection* problem is merely concerned with providing a binary answer to whether a given claim was verified before or not. The *retrieval* problem aims to find the actual verified claim for a given claim that is assumed to be previously-verified.

Accordingly, we propose a solution (denoted as *SpotVC*) that adopts a two-stage approach, detection and retrieval. The detection stage aims to build a *filter* that *efficiently* clears out claims that are not previously-verified before the expensive retrieval stage. The retrieval stage aims to build an *effective* search for relevant previously-verified claims *only if* the input claim is deemed as verified before during the detection stage.

For the detection phase, we propose different lightweight learning-based classifiers. Our experiments show that the proposed filter significantly outperforms threshold BERT-based classifiers and significantly saves computation time. For the retrieval stage, we develop a reranker that comprises a simple three-step pipeline. Our experiments show that the proposed reranker outperforms state-of-the-art (SOTA) models on two public datasets, and provides on-par performance on a third one. Overall, the end-to-end proposed system (*SpotVC*) integrates the filter and reranker components to demonstrate a good operational balance of efficiency and effectiveness for the real-world scenario.

In short, our contributions are four-fold:

- We introduce the verified claim detection problem. To address the problem, we propose several threshold BERT-based classifiers and exploit different feature categories to train lightweight learning-based classifiers.
- We construct and publicly share a new dataset for the verified claim detection task, with 1285 queries over 13,697 verified claims. We also release the source code of the experiments to facilitate reproducibility.[3]
- For the claim retrieval problem, we propose a preprocessing scheme and reranking step that outperform the state-of-the-art approaches on two public datasets and provide comparable performance on a third one.
- For the real-world scenario, our proposed end-to-end system significantly reduces the computation time while providing effective performance, exhibiting good operational balance.

Our presentation proceeds as follows. Section 2 details related work. Section 3 formally defines the problem we are addressing. Section 4 overviews *SpotVC* (our proposed solution) as a two-stage pipeline. Section 5 introduces our novel approach for addressing the verified claim detection sub-problem, then Section 6 illustrates the adopted methodology to tackle the retrieval sub-problem. In

[3] https://github.com/Watheq9/SpotVC.

Section 7, we investigate the possibility of using the proposed reranker to address the detection problem. Section 8 presents the full pipeline after integrating the filter and reranker parts. After discussing the implications of our study in Section 9, we finally present the conclusions and suggest future directions in Section 10.

2. Related work

The task of detecting previously verified claims attracted the attention of both research and academia. Looking at industry products, Google deployed a tool dubbed Fact Check Explorer.[4] This tool allows users to search in official fact-checking organizations only. The user can enter a query and view all previously fact-checked related claims from credible sources. However, this tool is not performing well with lengthy and sophisticated queries. Also, it has poor performance when it comes to the Arabic language.

From an academic perspective, there are two folds of research that target constructing retrieval systems for a given input query. The first tackles retrieving fact-checked articles for a given query. The fact-checking articles represent a long text that explains the claim, elaborates on its veracity, and justifies the verdict given by the fact-checkers. The second is just concerned with retrieving the statement of the verified claims. The statement of a verified claim may be up to four sentences long and summarizes the claim being discussed. Our work is mainly addressing this fold.

The verified claim retrieval task is categorized under the second fold. The task was proposed in Barrón-Cedeño et al. (2020), where a system is asked to retrieve all relevant previously verified claims for a given query, which is a claim-containing tweet. Since the verified claim retrieval task is the most similar to our work, we are reviewing the work performed in this direction.

2.1. Task emergence and datasets

Shaar et al. (2020) were the first to propose this task along with a related dataset. Shaar et al. (2020) adopts using BM25 (Robertson & Zaragoza, 2009) for the initial retrieval phase, then training RankSVM reranker using scores predicted by sentence-BERT and BM25. Following this, the task was proposed as a shared task in CheckThat! 2020 Lab (Barrón-Cedeño et al., 2020) in English language. The same task was proposed again with a more challenging dataset in CheckThat! 2021 Lab (Nakov et al., 2021) in two languages, English and Arabic. As a variation of this task, debate-based claim retrieval was proposed in 2021 Lab (Nakov et al., 2021; Shaar et al., 2021) as well. This variation is mainly concerned with retrieving previously fact-checked claims for a query that is mentioned in a political debate context. The organizers of CheckThat! Lab extended the 2021 dataset and proposed the task for the third time in 2022 (Nakov et al., 2022). At the time of writing, Hardalov et al. (2022) released a dataset with more than 330,000 pairs of tweets and verified claims with 10,340 unique verified claims from Snopes. However, this dataset contains many noisy examples since the pairs were collected from tweets containing a link to a fact-checking article.

2.2. Approaches

Many teams participating in CheckThat adopted a two-step pipeline; first, retrieving a set of documents and then applying a specific reranking approach over that set to optimize the effectiveness (Bouziane et al., 2020; Chernyavskiy et al., 2021; Passaro et al., 2020; Shliselberg & Dori-Hacohen, 2022; Thuma et al., 2020). Buster.Ai team (Bouziane et al., 2020), the winning team in the 2020 lab, fine-tuned RoBERTa model with adversarial hard negative samples to rerank pairs of tweets and verified claims. The strategy of Thuma et al. (2020) was to retrieve 1000 documents using the DPH retrieval model. Then, they extracted some features for those retrieved documents and trained LambdaMART. Aschern team (Chernyavskiy et al., 2021) (the 2021 winning team) followed nearly the same pipeline of Shaar et al. (2020) and Thuma et al. (2020) but with the difference that they added more features to train LambdaMART from a fine-tuned sentence-BERT. RIET Lab team (Top in CheckThat! 2022) (Shliselberg & Dori-Hacohen, 2022) utilized sentence-T5 (Ni et al., 2022) for the candidates' selection and GPT-Neo (Black et al., 2021) for the reranking.

Similar to Shaar et al. (2020), McDonald et al. (2020) extracted BM25 and cosine similarity scores for query-document pairs. After that, they trained some machine learning models to classify the pairs as relevant or not.

Differently, UNIPI-NLE team (Passaro et al., 2020), the second team in the 2020 lab, fine-tuned sentence-BERT in a cascaded way. First, they aimed to make the model assign a high cosine similarity score to the gold pairs. Then, they fine-tuned the sentence-BERT model to predict a relevance score for a tweet-verified-claim pair, then reranked pairs based on the relevance score.

2.3. Related tasks

Shaar et al. (2022) expanded the verified claim retrieval problem to handle documents. In short, for a given document, they extracted the sentences that can be verified by already verified claims, then retrieved the relevant verified claims for the extracted sentences. They approached the task by using RankSVM. Kazemi et al. (2021) deviated from the retrieval task and defined the claim matching task as the task of finding the pairs that can be served with one fact-check. Their focus was on finding matching claims between pairs of social media messages and pairs of fact-checks. The goal was to group the messages that state the same claim in different ways to help fact-checkers locate similar messages.

[4] https://toolbox.google.com/factcheck/explorer.

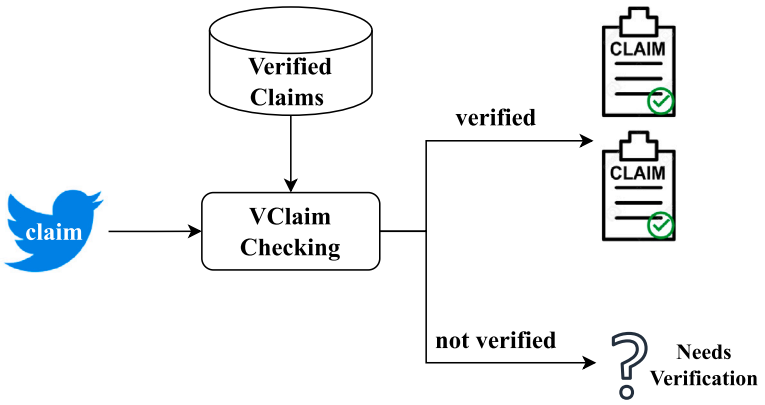


Fig. 2. Definition of the Verified Claim Checking problem.

To the best of our knowledge, all previous work assumes the presence of relevant claims for a particular query, despite the fact that this assumption does not reflect a real-world scenario. Besides, most of the aforementioned techniques did not pay much attention to the information value derived from URLs included in a tweet, converting the user handles to the corresponding usernames, or determining which variant of BERT is the best for this task. In this work, our goal is to fill in these gaps.

3. Problem definition

For fact-checkers, it is probable that a common claim on social media is just a repetition of an old viral claim that was already verified by a trusted fact-checking organization. As a result, it is worth developing an automated system that identifies those repeated claims. Such a system could reduce the needed time for verification, allowing more time to check the actual new claims.

We define the "Verified Claim Checking" (or *VClaim Checking* in short) problem as follows: given a check-worthy claim c presented in a tweet t_c and a collection of previously-verified claims V, the system provides a ranked list $R_c \in V$ of previously-verified claims that are (potentially) relevant to the claim c if it was indeed previously verified; otherwise, the system indicates that the claim c was not previously verified (hence it needs verification). The problem is illustrated in Fig. 2.

Although this definition reflects a real-world scenario, it was not fully addressed in the literature. All previous attempts dealt with the task as just a *ranking* problem, in which the system *always* retrieves relevant fact-checked claims, even if the input claim may not actually exist in the collection (i.e., not previously verified). In this case, the ideal system should return a "Not verified before" message to the user. In this work, we try to address this gap.

4. Approach overview

We approach the VClaim Checking problem by breaking it down into two sub-problems. The first sub-problem focuses on whether the given claim is verified before or not, viewed as a *classification* problem. We denote this sub-problem as *Verified Claim* ***Detection***. The second sub-problem assumes that the claim is already verified and focuses on retrieving the most relevant claims on top, viewed as a *ranked retrieval* problem. We denote this sub-problem as *Verified Claim* ***Retrieval***, following the same naming of 2020 and 2021 CheckThat! labs. The formal definition of these problems is as follows:

1. *Verified Claim Detection*: Given a check-worthy claim c presented in a tweet t_c and a set of previously-verified claims V, the system detects whether this claim was verified before or not.
2. *Verified Claim Retrieval*: Given a previously-verified check-worthy claim c presented in a tweet t_c and a collection of already-verified claims V, the system provides a ranked list R_c of verified claims that are potentially relevant to c.

Having introduced the problems, we propose *SpotVC*, our two-stage system architecture, that tackles them, which is illustrated in Fig. 3. The first stage acts as a *filter* that filters out non-verified claims. Since this filter will be, in reality, applied to a continuous stream of tweets containing claims, our main objective at this stage is *efficiency* while maximizing *accuracy* as much as possible. In fact, this filter enables the subsequent stage if it decides that the claim is already verified; otherwise, the system prompts the user that the claim was not previously verified and simply terminates. Our approach for that stage is detailed in Section 5.

Assuming the existence of some relevant claims for the input claim, the job of the *retrieval* stage is to identify those claims. At this stage, our main objective is maximizing *effectiveness*, i.e., we target retrieving the relevant claims at the top of the returned ranked list. Our approach for that stage is detailed in Section 6.

To that end, we present our proposed approach for the first and the second stages in Sections 5 and 6, respectively. Then in Section 7, we test the applicability of utilizing the best system proposed for the second stage (the reranker) as a solution for the first stage, highlighting the benefits brought by introducing the detection stage prior to retrieval and answering the intuitive question:

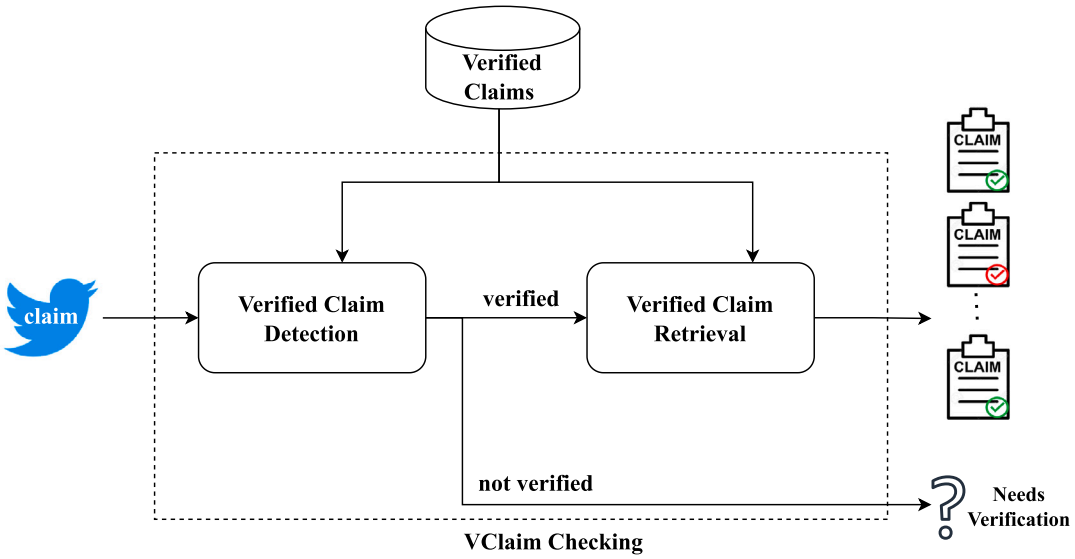


Fig. 3. The complete proposed system for addressing the verified claim problem.

Table 1
Overview of CT2022 and CT2022-D datasets.

Dataset	Train	Test	Collection size
CT2022	1285^{+}	322^{+}	13,835
CT2022-D	642^{+} + 643^{-}	161^{+} + 161^{-}	13,697

do we really need a filter for the VClaim Checking problem? For those sections, we present both the approach and the associated experimental evaluation in the same section. Finally, we combine the best out of the two stages and present the evaluation of the complete end-to-end system (*SpotVC*) in Section 8.

5. Verified claim detection

In this section, we introduce our methodology for addressing the verified claim detection problem. We first propose a new dataset for this particular task. Following that, we propose a novel lightweight learning-based method. Finally, we present the experimental results and discuss the main conclusion for this task.

5.1. Dataset creation

Since we are the first to propose the *verified claim detection* problem, we construct a new dataset that enables further research on this task. The new dataset is a modified version of CheckThat! 2022 English (CT2022) (Nakov et al., 2022). We recall that each query in CT2022 has at least one relevant fact-checked claim, which does not reflect the real-world scenario. Therefore, we modify CT2022 by randomly removing the relevant verified claims for 50% of the queries from both the collection and the gold labels. Therefore, we end up with a balanced dataset in which 50% of its queries are positive (i.e., having at least one relevant verified claim in the collection), and the other 50% queries are negative (having no relevant verified claims in the collection), making it a good fit for the proposed task. We denote this dataset as CT2022-D. Table 1 highlights the differences between CT2022 and CT2022-D.

5.2. Learning-based approach

In this subsection, we propose an efficient and lightweight approach that addresses the detection problem. Our proposed approach involves two steps. The first one involves extracting cheaply-computed features to train the classifier. In the second step, we train a lightweight machine-learning model and aim to maximize its effectiveness.

With respect to features, we select features from three main categories, namely classical/lexical retrieval, query performance predictors, and dense retrieval. While the retrieval weighting models capture the lexical match between the query and documents, query performance predictors help in estimating the quality of a retrieval model in case the relevant information is not available. Contrary to both, dense retrieval techniques depend on the semantic matching between the query and the documents. These categories provide different signals from different angles, hopefully supporting the right decision of the classifier as much as possible. Next, we present more details about the extracted features from each category.

- **Classical/Lexical Retrieval**: Generally, given a query, a retrieval model assigns a relevance score for a document, which is then used to rank the documents in descending order. We employ the scores produced by the classical/lexical retrieval models as features to train a machine learning model. For a given retrieval model, we extract three features: the maximum retrieval score, the minimum retrieval score, and the mean retrieval score of the retrieved list. The intuition behind choosing these values is attributed to the fact that they capture the range of the retrieval scores and their variation. Since there are a variety of lexical models in the information retrieval literature, we extract the features out of one model from each weighting family. We select BM25 (Robertson et al., 1995) as an IDF-based model, PL2 (Amati & Van Rijsbergen, 2002) as a Divergence from Randomness model, DLH (Amati, 2006) as a Hyper-geometric Divergence from Randomness model, Jelinek-Mercer (JM) smoothing (Jelinek, 1980) as a Language Model, and DFIZ (Dinçer et al., 2010; Kocabaş et al., 2014) as a Divergence from Independence model. Out of each weighting model, we extract the three features described above. However, we split this category of features into two sub-categories. The first one includes the scores from BM25 only and has only three features. We denote this category as *BM25*. The second sub-category contains the other four weighting models and involves 12 features. We denote this group as *Retr*. We opt to make BM25 as a standalone category because it is well-known in the literature as a powerful model.
- **Query Performance Predictors**: Query Performance Prediction (QPP) aims to predict the effectiveness of a retrieval system for unseen queries. QPP is categorized into two main groups: pre-retrieval QPP (PreQPP), which predicts the performance before the retrieval process, and post-retrieval QPP (PostQPP), which predicts it after retrieving documents. In general, for a given query, a query performance predictor assigns a score for that query, which indicates the difficulty of the query with respect to the retrieval system (Amati et al., 2004). The generated scores can provide meaningful signals for the classifier. In other words, hard queries should receive low scores from the predictors, making them indicative signals for the negative class, and vice versa. To that end, we adopt multiple query performance predictors from the literature while keeping in mind the efficiency constraint.
  We select three PreQPP predictors, namely the simplified clarity score (SCS) (He & Ounis, 2004), inverse document frequency (IDF) of query terms (He & Ounis, 2004; Plachouras et al., 2004), and collection-query similarity (SCQ) (Zhao et al., 2008). For IDF and SCQ, we followed the literature by computing the values as statistics (max, sum, mean, variance, and standard deviation). Thus, we end up with eleven PreQPP features. We also select the most common PostQPP predictors in the literature, namely Clarity (Cronen-Townsend et al., 2002), normalized query commitment (NQC) (Shtok et al., 2009, 2012), normalized standard deviation (NSD) (Cummins et al., 2011), and Weighted information gain (WIG) (Zhou & Croft, 2007). These predictors deal with general documents. However, the queries in this work are tweets, and the documents are verified claims. In other words, both documents and queries are short. Therefore, we need a predictor that takes this into consideration. Hasanain and Elsayed (2017) proposed multiple effective predictors that target search in microblogs. We add the best predictor from their study, which is Top Terms Coverage (TTC) (Hasanain & Elsayed, 2017). More specifically, we implemented four statistics from IDF-based TTC: median, mean, lower quartile, and upper quartile. As a result, we extracted eight features for PostQPP.
  Although there are many advancements in the QPP literature (Datta et al., 2022; Zamani et al., 2018), most of them depend on deep neural networks, which contradicts our goal of building a lightweight classifier. So, we avoid using these predictors as feature generators in our study.
- **Dense Retrieval**: The clear advantage brought by introducing dense retrieval to text ranking is removing the exact lexical match limitation. Dense retrieval can rank documents directly using their vector representations generated by transformers (Karpukhin et al., 2020; Xiong et al., 2020). This overcomes the lexical mismatch problem by capturing the semantic similarity between the query and the documents. Unlike monoBERT (Nogueira et al., 2019), which involves two steps, initial lexical-based retrieval and neural relevance-based reranking, dense retrieval involves only one step, which is performing the retrieval directly over documents.
  At inference, dense retrieval techniques are much more efficient than monoBERT for two reasons. First, in dense retrieval, the text representations of the documents are computed and stored in an offline mode, thus moving the expensive neural inference into that preprocessing step. Second, the inference step encompasses only the approximate nearest neighbor (ANN) search. We opt to study the effect of using the relevance scores generated by the dense retrieval model on the classification problem. To achieve that, we compute the min, max, and mean of the dense scores of the top k documents that are retrieved by the lexical retrieval model, e.g., BM25. To get those scores in our implementation, we run dense retrieval for the query and consider only the top k retrieved documents. Each document in the top k list of BM25 retrieval receives the corresponding DENSE score if it appears in the top k documents of the dense retrieval; otherwise, it receives a zero DENSE score.

Learning models To leverage the information from the aforementioned categories, we sought to train machine learning (ML) models that consider the efficiency constraint. Therefore, we select three commonly-used models for text classification, namely SVM, Logistic Regression, and XGBoost. The hyperparameters are tuned using randomized grid search cross-validation.

5.3. Experimental evaluation

Through our experiments for the detection problem, we attempt to answer the following research questions:

RQ1 How effective is the learning-based classifier? (Section 5.3.2)
RQ2 What is the best feature combination for the learning-based classifier? (Section 5.3.3)

Table 2
Performance of learning-based classifiers trained using all features on CT2022-D.

Model	Accuracy	F_1	Precision	Recall
XGBoost	**0.812**	**0.802**	0.848	0.761
Logistic regression	0.810	0.801	0.840	**0.766**
SVM	0.811	0.798	**0.857**	0.747

Table 3
The ablation study for different categories of features with respect to accuracy. The values starting from line 2 show the differences with reference to the baseline on line 1.

Line	Features	Time(XGBoost) in ms	Acc(XGBoost)	Acc(LR)	Acc(SVM)
1	All	293	0.812	0.810	0.811
2	All - {Retr}	−93	−0.001	0.004	−0.008
3	All - {Dense}	−45	−0.015	−0.017*	−0.019*
4	All - {PostQPP}	−123	0.004	−0.004	−0.005
5	All - {PreQPP}	−22	−0.001	**0.002**	−0.001
6	All - {BM25}	−28	**0.01**	0.001	−0.005
7	All - {PostQPP, Retr}	−216	0.002	0.001	−0.003
8	All - {PostQPP, Dense}	−171	−0.016	−0.015	−0.017
9	All - {PostQPP, PreQPP}	−144	0.004	0	**0.002**
10	All - {PostQPP, BM25}	−152	−0.001	0.001	−0.006
11	All - {PostQPP, Retr, Dense}	−261	−0.019*	−0.013	−0.019
12	All - {PostQPP, Retr, PreQPP}	−232	0.005	0	0
13	All - {PostQPP, Retr, BM25}	−229	−0.05*	−0.043*	−0.044*
14	All - {PostQPP, Retr, PreQPP, Dense}	**-275**	−0.02*	−0.015	−0.021*
15	All - {PostQPP, Retr, PreQPP, BM25}	−243	−0.052*	−0.043*	−0.042*

5.3.1. Experimental setup

Datasets In our experiments, we apply 5-fold cross-validation over our proposed dataset, CT2022-D. For reproducibility and fair comparisons, we maintain the same folds across all experiments.

Evaluation measures Since our problem is a binary classification problem and the data distribution is balanced, we choose four measures of performance, namely accuracy, precision, recall, and F_1 for the positive class. For each measure, we report the average performance over the five folds. We apply paired t-test for statistical significance testing over the accuracy measure only since we compute it *per query*; it is not possible to apply it for the other reported measures.

Features extraction and retrieval platform For learning-based classifiers, we first ran BM25 to retrieve an initial list of documents, then the features were extracted from that list. Additionally, we used PyTerrier (Macdonald & Tonellotto, 2020) for scoring the documents using multiple weighting models and for running ANCE dense retrieval (Xiong et al., 2020).

5.3.2. Learning-based classifiers (RQ1)

After extracting the features explained in Section 5.2, we trained SVM, Logistic Regression (LR), and XGBoost models on the five folds of CT2022-D. Table 2 shows the average performance of these models over the five folds. We observe that the three models exhibit close performance to each other, with a very slight superiority for XGBoost over accuracy and F_1.

Overall, the learning-based approach achieves a reasonable F_1 of 0.802 and an accuracy of 0.812.

5.3.3. Features ablation study (RQ2)

Although the results above show reasonable performance, it is not clear which features are the most and least informative for the detection task. Identifying such features can help find more efficient inference.

To achieve this goal, we conduct an ablation study in which we probe the importance of each feature category. The baseline is the model that is trained on all features categories, i.e., BM25, Retr, PreQPP, PostQPP, and Dense. The strategy of the ablation study is to remove one category of features at a time from the baseline, then check its effect on both effectiveness and efficiency. Afterward, we select the best combination in this stage for further reduction. The best combination is the one that received the least execution time, while its effectiveness is not statistically worse than the baseline. This process is repeated until there is no combination satisfying the previous condition.

Table 3 illustrates the results of the ablation study with respect to accuracy. Since the feature extraction time is substantially larger than the classifier's prediction time, the execution time for the three classifiers is nearly the same. Therefore, we report the execution time per query in milliseconds (Time pQ (ms)) for the XGBoost classifier only. The baseline in line 1 is trained on all available feature categories. The numbers in line 1 are just a repetition of the values in the first column of Table 2. From line 2 onwards, we are reporting the difference from the baseline in terms of time and accuracy.

In the first stage (lines 2–6), we noticed that eliminating PostQPP features led to the largest time gain while accuracy was not damaged significantly. In the second step (lines 7–10), we observed that adding Retr to the exclusion set gave additional time gain

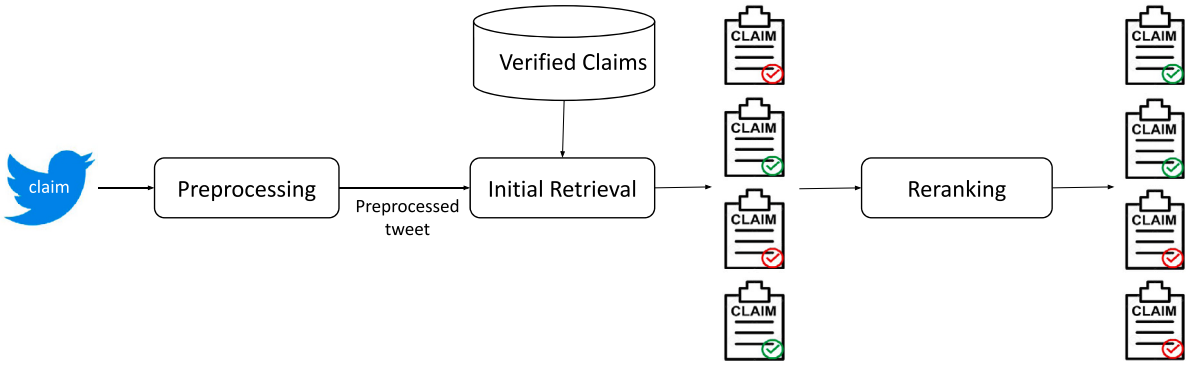


Fig. 4. The proposed pipeline for the verified claim retrieval task.

and a slight change in accuracy for all models. In the third stage (lines 11–13), we noted that removing PreQPP added more gain in time, and the accuracy witnessed a slight improvement for XGBoost but stayed the same for SVM and Logistic Regression. Another interesting observation is that excluding Dense or BM25 features harmed the accuracy significantly. In the last stage (lines 14–15), the classifier is trained on one category of features only (BM25 or Dense). Therefore, we noticed that accuracy saw a significant drop for nearly all models.

Overall, the best feature combination is BM25 and dense retrieval, i.e., line 12 of Table 3. This answers RQ2; BM25 and dense retrieval features are the most informative features for the learning-based classifiers.

6. Verified claim retrieval

In this section, we discuss our approach to addressing the retrieval problem, followed by the experimental setup and evaluation of its performance.

6.1. Approach

The retrieval problem implies two challenges. Firstly, informality is generally linked with tweets, and there is a lack of context due to the tweet's short length constraint. Secondly, claims might be paraphrased in different forms, suggesting a need for semantic matching.

To address these challenges, we propose a simple pipeline, shown in Fig. 4, comprising three steps.

1. **Preprocessing:** In this step, we eliminate noisy text patterns that might appear in the input tweet and expand it with potentially-useful information. In order to provide the tweets with additional context, we have expanded the following elements: URL links are converted into the titles of the corresponding Web pages, the mentions are replaced with the user names of the corresponding users, and the titles of the embedded images (similar to Bouziane et al. (2020)) or videos are obtained using reverse image search technique. More details can be found in our previous study (Mansour et al., 2022).
2. **Initial Retrieval:** The objective of this step is to get an initial set of retrieved claims and maximize the recall at the lowest possible cost. To achieve that, we incorporate an inexpensive retrieval weighting model, such as BM25 (Robertson et al., 1995), to acquire an initial set of claims. A tweet discussing a previously-verified claim is expected to have at least a few words in common with that claim. Consequently, it is quite likely that the relevant claims are present in the initial-retrieved set. However, the pertinent claims may be located farther down the list. The next reranking step addresses this problem.
3. **Reranking:** The final step of our proposed pipeline is employing a transformer-based reranker to rerank the initial-retrieved set. The primary goal of this step is to enhance effectiveness by utilizing semantic similarity to push the relevant claims to the top of the ranked list. To achieve that, we utilize monoBERT technique (Nogueira et al., 2019). MonoBERT is a point-wise learning-to-rank approach that utilizes BERT as a binary relevance classifier, by adding a two-node classification layer on top of BERT architecture. For a given pair of a query (tweet text) and a document (a previously verified claim), monoBERT model predicts a relevance score (expressing the relevance degree between the query and the document). Eventually, all documents in the initially-retrieved set will receive such a score and will be reranked accordingly. More details about the approach of constructing the training set and the fields used during training and inference can be found in Mansour et al. (2022).

6.2. Experimental setup

We followed the same setup and fine-tuning applied by Mansour et al. (2022), except that we conducted our experiments on CheckThat! 2021 English (CT2021) (Nakov et al., 2021) and CheckThat! 2020 English (CT2020) (Barrón-Cedeño et al., 2020) in addition to a third dataset, which is CheckThat! 2022 English (CT2022) (Nakov et al., 2022). CT2022 was constructed by expanding

Table 4
The effect of applying the proposed preprocessing (PreP) scheme on the initial retrieval performance using the development sets of CT2020, CT2021, and CT2022.

Model	CT2022			CT2021			CT2020		
	MAP@5	P@1	R@50	MAP@5	P@1	R@50	MAP@5	P@1	R@50
JM	0.854	0.817	0.970	0.879	0.844	0.955	0.687	0.558	0.919
JM+PreP	0.869	0.827	0.975	0.913⋆	0.884	0.965	0.712	0.579	0.944
BM25	0.860	0.822	0.970	0.889	0.864	0.955	0.710	0.594	0.914
BM25+PreP	**0.882**⋆	**0.847**	0.980	**0.922**⋆	**0.905**	0.965	**0.733**	**0.609**	**0.934**
DPH	0.818	0.757	0.965	0.868	0.834	0.955	0.685	0.563	0.919
DPH+PreP	0.848⋆	0.792	0.980	0.891⋆	0.849	0.970	0.721⋆	0.599	0.934
RM3	0.651	0.490	0.931	0.734	0.603	0.935	0.692	0.594	0.868
RM3+PreP	0.738⋆	0.614	0.965	0.758⋆	0.618	0.955	0.713	0.609	0.904

CT2021 with more recent queries. The adopted baselines are the top teams in each year, namely **Buster.AI** (Bouziane et al., 2020), **Aschern** (Chernyavskiy et al., 2021), and **RIET Lab** (Shliselberg & Dori-Hacohen, 2022) for CT2020, CT2021, and CT2022 respectively.

BERT variants Since there are several variants of BERT not utilized for verified claim retrieval, we opt to fill in this gap. That is, we employed the following BERT variants, S-MPNet and P-MPNet (Song et al., 2020),[5] RoBERTa (Liu et al., 2019),[6] Multinligual-MPNet (Song et al., 2020), and MiniLM (Wang et al., 2020).[7] In all experiments, hyperparameter values used for fine-tuning are as follows: learning rate (2e−5 or 3e−5), number of epochs (3, 4, or 5) and dropout rate (0.3 or 0.4).

Evaluation measures and significance test We applied the same evaluation measures proposed by CheckThat! organizers to perform a fair comparison with the baselines. As such, we opt for Mean Average Precision at depth 5 (MAP@5) as the main evaluation measure. We also report Precision@1 (P@1), Mean Reciprocal Rank (MRR), and Recall at 50 (R@50). As for the significance test, we applied paired t-test with Benjamini–Hochberg correction (Benjamini & Hochberg, 1995) to avoid the multiple comparisons problem. We add a star next to the values that satisfy the 5% significance level.

6.3. Experimental evaluation

For the verified claim retrieval problem, we mainly aim to answer the following research questions:

RQ3 Does the proposed preprocessing scheme enhance retrieval performance at the initial retrieval stage? (Section 6.3.1)
RQ4 What is the gain from exploiting BERT as a monoBERT reranker? Which variant of BERT is the most suitable for this task? (Section 6.3.2)

6.3.1. Initial retrieval with preprocessing (RQ3)

To answer RQ3, we measure the performance of multiple weighting models before and after applying the proposed preprocessing scheme over the three datasets, i.e., CT2020, CT2020, and CT2022. We namely ran Jelinek-Mercer (JM) smoothing (Jelinek, 1980), BM25 (Robertson & Zaragoza, 2009), RM3 (Lavrenko & Croft, 2001), and DPH (Amati et al., 2008). We employed PyTerrier (Macdonald & Tonellotto, 2020) for building indexes and performing the initial retrieval step.

Table 4 presents the evaluation results of those models before and after applying the proposed preprocessing scheme. It is clearly noted that all models received considerable improvements after applying the proposed scheme to the input queries. The improvements are consistent over the three datasets, and most of them are statistically significant. Nearly all models witnessed an improvement of about two points for all measures. Also, some models improved by three or four points, as we can see in MAP@5 over CT2021 for JM and BM25. These results clearly show that the proposed preprocessing pipeline provides significant improvements, and this answers RQ3.

Comparing models' performance after preprocessing, we found that BM25 is the best-performing model, receiving the highest values over all measures in all datasets. Therefore, we stick to using BM25 in all subsequent experiments.

6.3.2. MonoBERT for reranking (RQ4)

Toward answering RQ4, we first compare the performance of monoBERT against BM25, then experiment with multiple variants of BERT.

Performance gain Compared to BM25, we observed a noticeable gain in effectiveness by utilizing a monoBERT reranker (88% vs. 93% in MAP@5, and 85% vs. 90% in P@1 for CT2022 dev set). This exemplifies the efficacy that may be reached by utilizing contextualized models for this problem.

[5] We specifically use STSb-MPNet and Paraphrase-MPNet.
[6] https://huggingface.co/sentence-transformers/msmarco-roberta-base-v2.
[7] https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L12-v2.

Table 5
Performance of monoBERT rerankers on the test set of CT2020, CT2021, and CT2022. Significance differences with respect to the corresponding baseline are indicated by stars.

Model	CT2022			CT2021			CT2020		
	MAP@5	P@1	MRR	MAP@5	P@1	MRR	MAP@5	P@1	MRR
RIET Lab (Best at CT2022)	**0.956**	**0.943**	**0.957**	–	–	–	–	–	–
Aschern (Best at CT2021)	–	–	–	0.883	0.861	0.884	–	–	–
Buster.AI (Best at CT2020)	–	–	–	–	–	–	0.929	0.895	0.927
Roberta (CT2020 baseline)	0.920⋆	0.895⋆	0.922⋆	0.916	0.876	0.917	0.926	0.894	0.926
S-MPNet	0.944	0.928	0.945	**0.929**	**0.901**	**0.929**	**0.955**⋆	**0.950**⋆	**0.955**⋆
P-MPNet	0.923	0.900⋆	0.924⋆	0.922	0.886	0.923	0.944	0.925	0.944
Mul-MPNet	0.704⋆	0.589⋆	0.712⋆	0.742⋆	0.644⋆	0.749⋆	0.666⋆	0.553⋆	0.650⋆
MiniLM	0.911⋆	0.885⋆	0.913⋆	0.904	0.871	0.906	0.920	0.884	0.920

Best BERT variant Table 5 displays the results of our proposed system's evaluation throughout three distinct test sets, with the goal of comparing the proposed pipeline with the SOTA models and identifying the suitable BERT variant for this task. We highlight the statistical significant differences by stars with respect to the corresponding baseline. The following was gleaned from Table 5:

- S-MPNet is consistently the top-performing model among the models we experimented with over the three datasets. This is mainly attributed to the fact that it was trained on the Semantic Textual Similarity benchmark (STSb) (Cer et al., 2017). This benchmark shares some similarities in nature with the task we are addressing.
- Compared with the baselines, S-MPNet outperformed the baselines of CT2020 and CT2021, with a statistical difference in CT2020 over all measures. With respect to CT2022, the difference between S-MPNet and the baseline is only 1.2 point, with no statistical difference. It is vital to mention that RIET Lab's method is very expensive in terms of computations, as it requires sentence-T5 (Ni et al., 2022) for candidate selection and GPT-Neo (Black et al., 2021) for reranking the candidate list. While sentence-T5 involves 110M up to 11B parameters (according to the used model), GPT-Neo comes with 1.3B parameters. As a result, their proposed model requires 1.41B parameters at the cheapest setup, while our proposed reranker needs only 110M parameters. In other words, the model used by RIET Lab is more than 12 times bigger than ours, and the difference in effectiveness is insignificant.

In summary, the proposed pipeline outperformed the SOTA models on CT2020 and CT2021 and showed comparable performance on CT2022. This indicates the high effectiveness of this pipeline.

7. BERT reranker for detection

In this section, we demonstrate how a monoBERT reranker can be utilized as a classifier for solving the verified claim detection problem. We first propose three BERT- and threshold-based classifiers, then we evaluate and compare their performance to the previously-described learning-based classifiers.

As explained earlier, monoBERT assigns a relevance score for each document; then, documents are reranked based on those scores. Based on this fact, we propose three filters that employ those scores:

- **Relevance-based Classifier (RbC)**: This classifier simply considers only the relevance score predicted by the reranker for the top document; if the score is greater than a target threshold τ (a hyper-parameter), then it predicts a positive query; otherwise, it predicts a negative query. The intuition behind that classifier is that if there is at least one relevant claim for the input query, it should be the top document. Therefore, its relevance score might help decide whether there exists a relevant verified claim for the query (positive) or not (negative).
- **Diff-based classifier (DbC)**: This classifier considers only the difference between the relevance scores of the top two documents. If the difference is greater than a target threshold τ (a hyper-parameter), then it predicts a positive query; otherwise, it predicts a negative query. This classifier is helpful when there is, at most, one relevant verified claim for the input query.
- **Entropy-based Classifier (EbC)**: Instead of considering only the top retrieved document or two, this classifier computes the entropy of the predicted relevance scores of the top k documents to get a sense of the distribution of those scores. If the entropy is smaller than a target threshold τ (a hyper-parameter), then it predicts a positive query; otherwise, it predicts a negative query. The low entropy value indicates high information gain in the relevance scores, which we interpret as an indication of the presence of relevant claims.

7.1. Experimental evaluation

In the following experiments, we target answering the subsequent research questions:

RQ5 Can monoBERT reranker be effectively used as a threshold-based classifier? (Section 7.1.1)
RQ6 Does the best proposed learning-based classifier outperform the best threshold-based classifier? (Section 7.1.2)

Table 6
Performance of threshold-based classifiers using 5-fold cross validation.

Model	Threshold	Accuracy	F_1	Precision	Recall
EbC	0.836	0.611	0.677	0.577	0.821
RbC	0.824	**0.783**	**0.808**	**0.726**	**0.912**
DbC	0.302	0.620	0.630	0.612	0.649

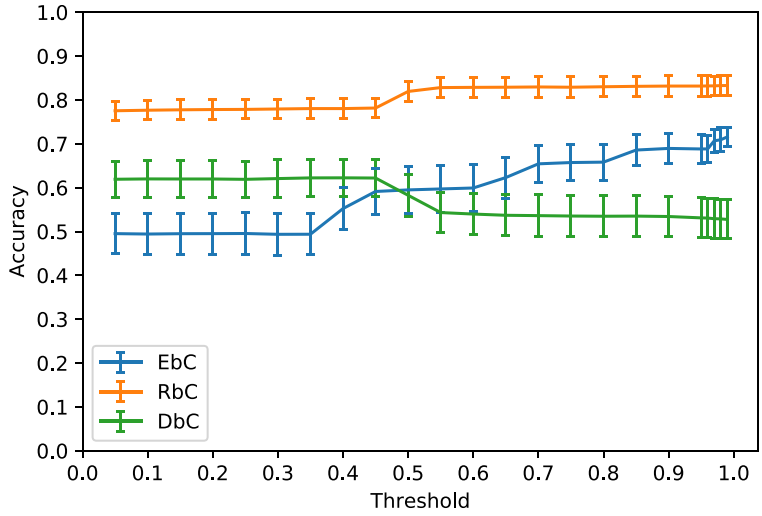


Fig. 5. Sensitivity of threshold-based classifier to changes in the threshold value. The error bar represents the standard deviation of accuracy over five folds.

7.1.1. Threshold-based classifiers (RQ5)

To get the relevance scores, we utilized the best monoBERT reranker attained in Section 6.3.2, namely S-MPNet. For tuning the thresholds, we used *nested* 5-fold cross-validation so that each threshold is tuned on the inner dev fold(s) on values ranging from 0.05 to 0.95 with 0.05 steps and from 0.95 to 0.99 with 0.01 steps. The best threshold value is then used to report the performance on the test fold. The final reported threshold value is simply the average of the best five. We used Macro F_1 measure for tuning the thresholds.

Table 6 shows the evaluation results of the three proposed threshold-based classifiers. We notice that the RbC outperforms the other classifiers by a large margin, which suggests that the relevance score of the top document is a highly informative feature.

To further analyze the performance of the threshold-based classifiers and check their sensitivity to changing the threshold value, we computed the average and standard deviation of accuracy for each threshold over the dev folds. Fig. 5 illustrates the effect of changing the threshold value on the classifier's accuracy. The error bars represent the standard deviation (95% confidence intervals) of accuracy. The figure shows clearly that RbC exhibits stable performance (i.e., is less sensitive) with threshold values above 0.5. It also indicates that RbC is by far the best-performing model over all experimented threshold values.

In short, the takeaway message that answers RQ5 is that a monoBERT reranker can be used as a threshold-based classifier for addressing the detection problem. Additionally, among the different models, RbC, i.e., the one that depends on the relevance score for the top document only, is the best.

7.1.2. Threshold-based vs. Learning-based classifiers (RQ6)

Having identified the best threshold-based classifier and the best feature combination for a learning-based classifier (in Section 5.3.3), we draw a comparison between them. In particular, we compare the performance between RbC and the learning-based classifiers after training them on Dense and BM25 features. We consider RbC as a baseline in statistical significance tests over accuracy.

We illustrate the evaluation results in Table 7. It can be clearly witnessed that learning-based classifiers outperformed RbC over all measures except for recall. This conclusion is of utmost importance as it means the learning-based classifiers can *efficiently* answer the detection problem without the need to run a costly model like BERT.

Among the learning-based classifiers, XGBoost exhibits the best performance for all measures except precision. Therefore, we choose this model for subsequent experiments. Putting the best filter and the best retrieval system in one full pipeline is examined in the next section.

Table 7
Evaluation of ML classifiers trained using BM25 and Dense features only.

Models	Accuracy	F_1	Precision	Recall
RbC	0.783	0.808	0.726	**0.912**
XGBoost	**0.817**⋆	**0.810**	0.843	0.78
Logistic regression	0.810⋆	0.800	0.839	0.766
SVM	0.811⋆	0.799	**0.855**	0.751

8. End-to-End system

This section demonstrates how we integrate the detection and retrieval models together in an end-to-end system to solve the verified claim checking problem. We also model the efficiency computation and illustrate how the effectiveness of the *full* system is estimated. Finally, we present the evaluation results and elaborate on the effectiveness and efficiency trade-off.

The usage scenario of the system is as follows: A claim-containing tweet is preprocessed and passed as input to the filter. The filter decides whether the claim has been previously-verified or not. The "*no*" answer terminates the process and results in a message indicating that. The "*yes*" answer enables the retrieval stage, which provides the user with a list of potentially-relevant verified claims.

We develop the full system based on the conclusions attained from Sections Section 5, 6, and 7. Aligned with the system shown in Fig. 3, the filter is represented by an XGBoost classifier trained on two categories of features, namely BM25 and Dense, while the retrieval component is represented by the MPNet reranker. We denote the proposed full system as *SpotVC*.

That being said, the system needs to be evaluated in terms of both effectiveness and efficiency. While measuring efficiency can be done through execution time, evaluating effectiveness requires special handling. Following this, we elaborate on the procedure we followed to perform such an evaluation. All experiments are performed using the CT2022-D dataset.

8.1. Modeling efficiency

In our proposed system (*SpotVC*), there are two major components, namely, the filter and the reranker. For a given input, *SpotVC* might either run the filter only if it sees the claim was not previously-verified, or the full pipeline (filter + reranker) otherwise. To account for both cases, we opt to compute the time needed for input queries that are *not* previously verified (denoted as *not verified queries or* Q_{nv}), and for input queries that are previously-verified (denoted as *verified queries* or Q_v) per the gold labels in our test collection. We then estimate the overall average response time as follows:

$$t = \alpha * t_v + (1 - \alpha) * t_{nv} \tag{1}$$

where t is the average response time of *SpotVC*(the full system), t_v is the average response time for the *verified queries*, t_{nv} is the average response time for the *non-verified queries*, and α represents the percentage of the *previously-verified queries* in the test set. Eq. (1) represents a crude attempt to model the system's efficiency. t_v and t_{nv} are computed as follows.

$$t_v = \frac{1}{n_v} \sum_{q \in Q_v} t(q) \qquad t_{nv} = \frac{1}{n_{nv}} \sum_{q \in Q_{nv}} t(q) \tag{2}$$

where $t(q)$ is the response time of the system given the query q, which can either be the filter time only or the time of both the filter and the reranker, depending on the filter's prediction.

When most of the queries are detected as previously-verified, the reranker's usage rate will increase substantially, which means the reranker time will mostly dominate the response time. Otherwise, there will be less dependency on the reranker. The main focus here is on the usage frequency of the reranker, as the filter is used either way, and the reranker is the most expensive component in the proposed system.

8.2. Modeling effectiveness

Since we are the first to propose the task described in Section 3, we introduce the suitable evaluation measure. The proposed measure is simply a modified version of MAP@5, denoted as MAP@5⋆. The modification is inspired by TREC-2015 Microblog Track (with a real-time filtering task) (Lin et al., 2015). Lin et al. (2015) modified the normalized cumulative gain (nCG) measure so that it takes into consideration the days in which there are no actual relevant tweets to return to the user. Similarly, we account for the queries that do not have relevant verified claims in the collection.

The system is evaluated according to the vanilla MAP@5 measure only if the query is previously-verified *and* the filter predicts 1 to enable the retrieval stage. Otherwise, the system is either fully rewarded (with a score of 1) if the filtering prediction is correct or fully penalized (with a score of 0) if the filtering prediction is incorrect.

To model the average effectiveness of *SpotVC*, we followed the same strategy we adopted for efficiency in Section 8.1, replacing the response time values with the MAP@5* effectiveness values for each query in Eqs. (1) and (2).

8.3. Experimental evaluation

In this section, we aim to answer the following research questions:

RQ7 How efficient would the proposed pipeline be? How far is it from the ideal system? (Section 8.3.2)
RQ8 What is the effectiveness of the proposed system? How far is it from the ideal system? (Section 8.3.2)

8.3.1. Experimental setup

Hardware configuration All experiments were conducted on a Linux machine running Intel Xeon E5-2690 v4 CPU, Tesla P100 16 GB GPU, and 128 GB RAM.

Average detection time The time required to run the filter (XGboost classifier) is indicative of how long it will take to extract BM25 and Dense features and provide a classification prediction; the total time is simply the sum of them. To get an accurate estimation of the execution time, we repeated the experiment five times and computed the average. We found that the average time for running the filter per query is **61** ms.

Average reranking time As mentioned earlier, the retrieval stage only involves the reranker since the initial retrieval phase is done during the filtering process. Consequently, the retrieval time is the time taken by the reranker (MPNet reranker). To achieve this, we repeated the experiment five times and reported the average. We realized that the reranker required, on average, **1278** ms per query.

Baselines Since we are the first to introduce the Verified Claim Checking problem, we opt to compare our proposed system against two baselines, namely the *reranker only* and the *ideal* systems defined as follows:

- **Retrieval Only (denoted as RO)** is merely the retrieval system that encompasses the initial retrieval and reranking phases without using the filter, i.e., a system that skips the filtering stage. The rationale is to validate the value of the proposed filter component. The average total time for running the full retrieval stage is **1292** ms.
- **Ideal (denoted as IDEAL)** refers to the perfect system we aim to approach. It comprises an ideal filter and an ideal reranker, where the filter's decisions are always correct, and the relevant verified claims are perfectly retrieved. The effectiveness of IDEAL is, intuitively, always 1. As for efficiency, if the query is not previously verified, then the execution time is merely the filtering time; otherwise, it is the filtering plus reranking times. Consequently, the execution time is 61 ms for Q_{nv} and 1339 ms for Q_v. By contrasting our proposed approach against **IDEAL**, we gauge the goodness of our system.

α ***Variations*** To evaluate the performance of our proposed system in different scenarios of non-verified/verified claims, we computed the effectiveness and efficiency values according to Eq. (1) while varying α from 0.0 to 1.0 with 0.05 increments. We apply this strategy for both *SpotVC* and the baselines.

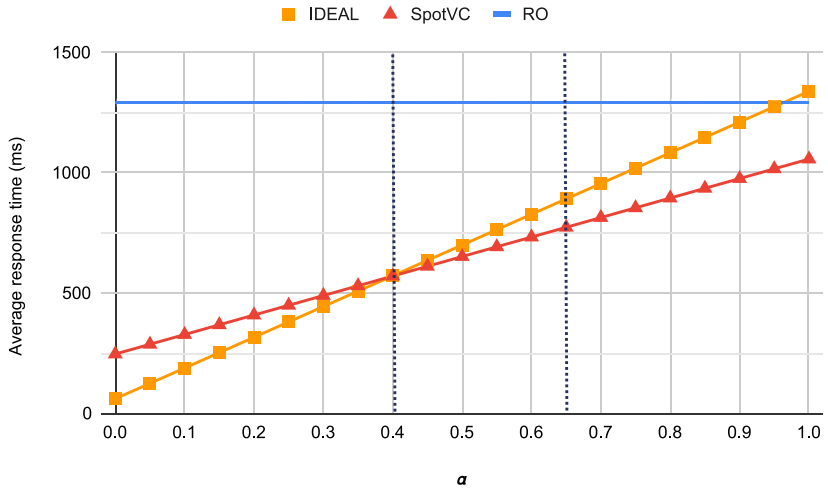
8.3.2. Experimental results

Efficiency (RQ7) Fig. 6(a) compares the efficiency of *SpotVC* against *IDEAL* and *RO*. In response to RQ7, it is evident that our proposed system *SpotVC* is *always* more efficient than RO. This clear efficiency improvement over all α values reveals the enormous boost brought on by the proposed filter component. In the same breath, this observation addresses an intuitive inquiry "Why don't we just use the reranker directly?". Running a reranker is a costly process, and our goal is to avoid it when it is not needed. We also notice that as α goes beyond 0.40, *SpotVC*'s efficiency outperforms *IDEAL*'s efficiency. This is mainly attributed to the increasing number of false negatives. This, of course, harms the effectiveness, as we see next.

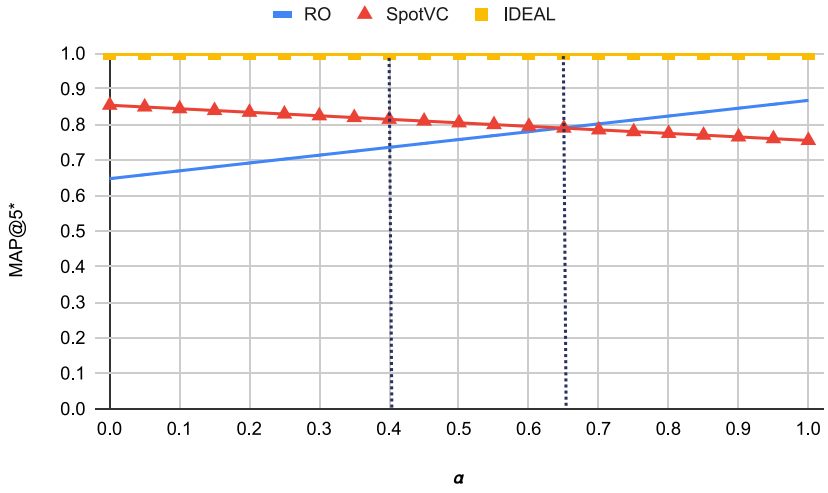
Effectiveness (RQ8) Fig. 6(b) demonstrates the effectiveness of *SpotVC*, *RO*, and *IDEAL* systems across different values of α with respect to MAP@5*. We note that *SpotVC* outperforms *RO* when α is less than 0.65 while exhibiting a performance close to *IDEAL* for very small values of α. Recall that small values of α indicate a low percentage of previously-verified claims. Additionally, *SpotVC* still delivers reasonable performance even for higher values of α since nearly all MAP@5* values are in the upper quarter. In fact, MAP@5* ranges from 0.86 when α is zero to 0.73 when α is one. Though *SpotVC* is considerably closer to *IDEAL* for small values of α, the difference becomes bigger for large values of α. That is, the higher the percentage of *previously-verified* queries, the lower the effectiveness of the proposed system. This is attributed to the increased number of false negatives predicted by the filter. That emphasizes the importance of acquiring higher accuracy for the filter, which is one of the potential future directions.

Effectiveness and efficiency trade-off Having discussed the effectiveness and efficiency of the proposed system, we turn to discuss the trade-off between them. We discuss the effect of changing α on effectiveness and efficiency across three ranges.

- $\alpha \in [0.0 - 0.4]$: Intuitively, in the real-world scenario, the number of tweets stating previously-verified claims is very small compared to those stating new claims (that are not previously-verified). Therefore, we expect α to be *very small* in the real-world application. This range ([0.0−0.4]) then gives an indication of the proposed system's performance in real-world use cases. Since there is no previous work that measured this percentage, let us consider an extreme case and assume α to be 20%, i.e., out of every 5 new tweets containing claims, one tweet mentions a previously-verified claim. Looking at Fig. 6 when α is 0.2, *SpotVC* clearly outperforms *RO* in both efficiency and effectiveness, while not very far from *IDEAL*, especially in terms of efficiency.
- $\alpha \in [0.4 - 0.65]$: In this mid-range, *SpotVC* is still better than *RO* in both efficiency and effectiveness. However, it starts to diverge a bit from *IDEAL* in effectiveness while beating it in efficiency.



(a) Efficiency



(b) Effectiveness

Fig. 6. Full system evaluation.

- $\alpha \in [0.65 - 1.0]$: In this range, *SpotVC* becomes the most efficient system; however, this harms its effectiveness due to the increased number of false negatives predicted by the filter. In fact, *RO* outperforms *SpotVC* in terms of effectiveness only in this range. It is worth noting here that while this range is theoretically possible, it is not practically expected, as we discussed earlier.

In short, *SpotVC* shows a good balance of effectiveness and efficiency for small values of α (even in its mid-range), which we expect to cover real-world use, and better efficiency at the expense of effectiveness for large values of α.

Error analysis of filtering As the filtering phase is critical and has a crucial impact on the overall performance of the end-to-end system, Table 8 lists some failure cases of the best filter (XGBoost classifier trained on BM25 and DENSE features). In particular, we present one false positive case (query #1) and one false negative case (query #2). Our discussion depends merely on the involved features in the filter's decision, which are BM25 and DENSE features (min, max, and mean for each). Recall that the features are only computed for the documents retrieved in the initial retrieval phase, i.e., those retrieved by the BM25 retrieval model. This means the DENSE score will be zero if the document does not appear in the top k retrieved DENSE list. To that end, for query #1, we can clearly see a good lexical overlap between the query and the top documents retrieved by BM25 model, and two (in bold) out of the top 3 do exist in the DENSE retrieved list. This means the scores for BM25 and DENSE are relatively high, encouraging the filter to wrongly predict a positive query. Concerning query #2, although the relevant verified claim is at rank 1 in BM25 retrieved list, none of the documents retrieved by BM25 appeared in the DENSE retrieved list; consequently, all documents received zero

Table 8
Sample of misclassified examples by XGBoost filter. The lexical overlap is highlighted in yellow. The verified claims which also appear in the DENSE's retrieved list are in **bold**. The relevant verified claim is in *italic*, if any. Scores between parenthesis at the end of the verified claim indicate the BM25 and DENSE scores, respectively.

Query	Top retrieved verified claims by BM25
1. WATCH: President Trump met with loud boos as he is introduced at the World Series in Nationals Park on Sunday night., social group, NBC News ( ) October 28, 2019	1. **U.S. President Donald Trump tweeted about banning "fake boos" after being booed at the 2019 World Series.** (43.57, 715.39) 2. Watch Night church services began in 1862 with blacks in the U.S. awaiting the enactment of the Emancipation Proclamation of New Year's Day, 1863. (26.05, 0.0) **3. More people watched President Trump's 2019 State of the Union address on television than watched Super Bowl Super Bowl LIII.** (24.53, 714.78)
2. Honest question the media should ask Nancy Pelosi Elijah E. Cummings – Did you have a fake impeachment presser and then fly off to Venice? cc: Juliegrace Brufke Manu Raju Daily Callerbrooke FoxNews Nicholas Fandos Jeremy Herb — Matt Gaetz ( ) July 27, 2019	1. *In July 2019, U.S. Rep. Elijah Cummings was in Venice, Italy, with other Democratic lawmakers.* (41.16, 0.0) 2. U.S. Rep. Matt Gaetz, R-Florida, stated, "We better use" the Second Amendment when railing against alleged suppression of speech by social media platforms. (36.07, 0.0) 3. In May 2019, a conference of U.S. governors voted to impeach U.S. House Speaker Nancy Pelosi, setting the table for her removal from office. (32.77, 0.0)

DENSE scores, making the DENSE feature values extremely low (or zero). This gives a strong indication to the filter that this query has no relevant claims, causing a false negative case. In fact, upon our investigation, we found out that this same reason is behind many false negative cases, suggesting the need for enhancing the DENSE score computations and the retrieval model in general. One possible enhancement is to compute exact DENSE scores for each document in the BM25 retrieved list (instead of assigning a zero score for documents that do not appear in the top k DENSE list). Another direction is to fine-tune the DENSE model on the task at hand. We leave that to future work.

9. Implications of our study

In this section, we outline the theoretical and practical implications of our study.

9.1. Theoretical implications

Several studies have attempted to address the verified claim checking problem, but, to the best of our knowledge, all see it as a ranking problem that implies that the input claim is already previously-verified, hence the existence of some claims that are relevant to it. In this study, we bridge this gap by proposing a system and a suitable dataset. Our work has the following theoretical implications:

- **Drawing attention to the missing piece of previous research on the verified claim retrieval problem:** We shed light on the significance of responding to claims that were not previously verified. We demonstrate the value of having a system capable of addressing a frequent real-world problem.
- **Open new doors for researchers to work on the task:** We propose a new dataset and make it publicly available to enable future research on performing further improvements.

9.2. Practical implications

Since the verified claim checking problem could be intimidating for everyone in the community, the proposed system can be used by journalists to assist them in preparing reports, by fact-checkers while inspecting new viral claims, and by ordinary social media users who desire to verify certain news. As such, the practical implications of the proposed work are as follows:

- **Providing an essential community service:** We envision that the proposed system could be integrated into a web API service, which could subsequently be utilized by mobile applications or websites for labeling incoming tweets. Such a service will facilitate the usage for social media users, therefore mitigating the detrimental effects of fake news.
- **Developing a tool that saves time for fact-checkers and journalists:** As journalists and fact-checkers might need to check if a specific claim has been verified before, a real-time response may save a considerable amount of time for those types of users.
- **Applicability to other tasks and social media platforms:** Our proposed solution to the Verified Claim Checking problem could be easily applied to other similar tasks. One example is the task of detecting previously-answered questions in community question–answering (CQA) websites, such as Stack Overflow or Quora. Integrating *SpotVC* in CQA systems could substantially help in marking the duplicated questions and retrieving the relevant previously-answered questions for a user's query. Additionally, despite being designed to work on tweets, *SpotVC* can be applied to other social media platforms, such as Facebook and Instagram.

10. Conclusion and future work

In this paper, we have introduced the Verified Claim Checking problem to match the real-world scenario. To address the problem, we proposed a two-stage approach. The first phase aimed at an *efficient* filtering of non-previously-verified claims. We created a new dataset for that task and found that BM25 and dense retrieval features provided both effective and efficient learning-based filters that outperformed the BERT-based classifiers. The second stage aimed at an *effective* retrieval of previously-verified claims. Our experiments demonstrated the superiority of our simple proposed reranker over the SOTA models on two datasets and the on-par performance with a SOTA model on a third one. Furthermore, we investigated the performance of the full end-to-end pipeline to address the real-life scenario. Our analysis showed that *SpotVC* filtering stage provided a significant time gain compared to a retrieval-always approach, proving the need for such a design. Overall, *SpotVC* is very effective in retrieving relevant previously-verified claims while exhibiting an efficient performance that is not very far from ideal in the practical real scenario.

Our work opens the door for several research directions, e.g., improving the filter's accuracy, applying the proposed pipeline to other languages, and deploying *SpotVC* in the real world. The latter, in particular, can support an online model update through real-time user feedback in addition to a periodic update of the collection of previously-verified claims that is vital to consider newly-spreading claims.

CRediT authorship contribution statement

Watheq Mansour: Designed the ideas and research goals, Constructed the data, Designed and developed the methodology, Designed the research questions and the experiments, Developed and implemented the system, Conducted the experiments, Evaluated the performance, Applied statistical tests, Investigated and analyzed the results, Writing – orginal draft, Writing – Review & Editing. **Tamer Elsayed:** Supervision, Acquired the fund for the project, Designed the data construction approach, Devised the detection problem, Designed the research questions and the experiments, Provisioned required hardware, Analyzed the results, Writing – Review & Editing. **Abdulaziz Al-Ali:** Supervision, Acquired the fund for the project, Designed the research questions and the experiments, Analyzed the results, Writing – Review.

Data availability

The dataset is available over this repository: https://github.com/Watheq9/SpotVC.

Acknowledgments

This work was made possible by NPRP grant# NPRP11S-1204-170060 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Amati, G. (2006). Frequentist and bayesian approach to information retrieval. In *European conference on information retrieval* (pp. 13–24). Springer.

Amati, G., Amodeo, G., Bianchi, M., Gaibisso, C., & Gambosi, G. (2008). FUB, IASI-CNR and University of "Tor Vergata" at TREC 2008 Blog Track. In *The seventeenth text retrieval conference, TREC 2008*. US.

Amati, G., Carpineto, C., & Romano, G. (2004). Query difficulty, robustness, and selective application of query expansion. In *European conference on information retrieval* (pp. 127–137). Springer.

Amati, G., & Van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 357–389.

Barrón-Cedeño, A., Elsayed, T., Nakov, P., Da San Martino, G., Hasanain, M., Suwaileh, R., Haouari, F., Babulkov, N., Hamdan, B., Nikolov, A., Shaar, S., & Ali, Z. (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* (pp. 215–236).

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *57*(1), 289–300.

Black, S., Leo, G., Wang, P., Leahy, C., & Biderman, S. (2021). GPT-Neo: Large scale autoregressive language modeling with Mesh-Tensorflow. http://dx.doi.org/10.5281/zenodo.5297715.

Bouziane, M., Perrin, H., Cluzeau, A., Mardas, J., & Sadeq, A. (2020). In L. Cappellato, C. Eickhoff, N. Ferro, & A. Névéol (Eds.), *CEUR workshop proceedings*, *Buster.AI at CheckThat! 2020: Insights and recommendations to improve fact-checking*. CEUR-WS.org.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (pp. 1–14).

Chernyavskiy, A., Ilvovsky, D., & Nakov, P. (2021). In G. Faggioli, N. Ferro, A. Joly, M. Maistro, & F. Piroi (Eds.), *CEUR workshop proceedings*, *Aschern at CheckThat! 2021: lambda-calculus of fact-checked claims*. CEUR-WS.org.

Cronen-Townsend, S., Zhou, Y., & Croft, W. B. (2002). Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 299–306).

Cummins, R., Jose, J., & O'Riordan, C. (2011). Improved query performance prediction using standard deviation. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval* (pp. 1089–1090).

Datta, S., Ganguly, D., Greene, D., & Mitra, M. (2022). Deep-qpp: A pairwise interaction-based deep learning model for supervised query performance prediction. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 201–209).

Dinçer, B. T., Kocabas, I., & Karaoglan, B. (2010). IRRA at TREC 2010: Index term weighting by divergence from independence model. In *TREC*.

Hardalov, M., Chernyavskiy, A., Koychev, I., Ilvovsky, D., & Nakov, P. (2022). CrowdChecked: Detecting previously fact-checked claims in social media. In *Proceedings of the 2nd conference of the asia-pacific chapter of the association for computational linguistics and the 12th international joint conference on natural language processing* (pp. 266–285).

Hasanain, M., & Elsayed, T. (2017). Query performance prediction for microblog search. *Information Processing & Management*, *53*(6), 1320–1341.
He, B., & Ounis, I. (2004). Inferring query performance using pre-retrieval predictors. In *International symposium on string processing and information retrieval* (pp. 43–54). Springer.
Jelinek, F. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proc. workshop on pattern recognition in practice, 1980*.
Kang, C., & Goldman, A. (2016). In Washington pizzeria attack, fake news brought real guns. *New York Times*, *5*.
Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 6769–6781).
Kazemi, A., Garimella, K., Gaffney, D., & Hale, S. (2021). Claim matching beyond English to scale global fact-checking. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing* (pp. 4504–4517).
Kocabaş, İ., Dinçer, B. T., & Karaoğlan, B. (2014). A nonparametric term weighting method for information retrieval based on measuring the divergence from independence. *Information Retrieval*, *17*(2), 153–176.
Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 120–127).
Lin, J. J., Efron, M., Sherman, G. T., Wang, Y., & Voorhees, E. M. (2015). Overview of the TREC-2015 microblog track. In *TREC*.
Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
Macdonald, C., & Tonellotto, N. (2020). Declarative experimentation in information retrieval using PyTerrier. In *Proceedings of ICTIR 2020*.
Mansour, W., Elsayed, T., & Al-Ali, A. (2022). Did I see it before? Detecting previously-checked claims over Twitter. In *European conference on information retrieval* (pp. 367–381). Springer.
McDonald, T., Dong, Z., Zhang, Y., Hampson, R., Young, J., Cao, Q., Leidner, J., & Stevenson, M. (2020). In L. Cappellato, C. Eickhoff, N. Ferro, & A. Névéol (Eds.), *CEUR workshop proceedings*, *The university of sheffield at CheckThat! 2020: claim identification and verification on twitter*. CEUR-WS.org.
Nakov, P., Da San Martino, G., Alam, F., Shaar, S., Mubarak, H., & Babulkov, N. (2022). *Overview of the CLEF-2022 CheckThat! lab task 2 on detecting previously fact-checked claims*: *Working Notes of CLEF*.
Nakov, P., Giovanni, D. S. M., Elsayed, T., Barrón-Cedeño, A., Míguez, R., Shaar, S., Alam, F., Haouari, F., Hasanain, M., Mansour, W., Hamdan, B., Ali, Z. S., Babulkov, N., Nikolov, A., Shahi, G. K., Struß, J. M., Mandl, T., Kutlu, M., & Kartal, Y. S. (2021). Overview of the CLEF-2021 CheckThat! lab on detecting check-worthy claims, previously fact-checked claims, and fake news. In *CLEF 2021*, *Proceedings of the twelfth international conference of the CLEF association: experimental IR meets multilinguality, multimodality, and interaction*.
Ni, J., Abrego, G. H., Constant, N., Ma, J., Hall, K., Cer, D., & Yang, Y. (2022). Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the association for computational linguistics: ACL 2022* (pp. 1864–1874).
Nogueira, R., Yang, W., Cho, K., & Lin, J. (2019). Multi-stage document ranking with BERT. arXiv abs/1910.14424.
Passaro, L., Bondielli, A., Lenci, A., & Marcelloni, F. (2020). In L. Cappellato, C. Eickhoff, N. Ferro, & A. Névéol (Eds.), *CEUR workshop proceedings*, *UNIPI-NLE at CheckThat! 2020: Approaching fact checking from a sentence similarity perspective through the lens of transformers*. CEUR-WS.org.
Plachouras, V., He, B., & Ounis, I. (2004). University of glasgow at TREC 2004: Experiments in web, robust, and terabyte tracks with terrier. In *TREC*.
Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. In *Overview of the third text retrieval conference (TREC-3)* (pp. 109–126). Gaithersburg, MD: NIST, URL: https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.
Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333–389.
Shaar, S., Alam, F., Martino, G. D. S., & Nakov, P. (2022). Assisting the human fact-checkers: Detecting all previously fact-checked claims in a document. In *Proceedings of the 2022 conference on empirical methods in natural language processing*.
Shaar, S., Babulkov, N., Da San Martino, G., & Nakov, P. (2020). That is a known Lie: Detecting previously fact-checked claims. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 3607–3618).
Shaar, S., Haouari, F., Mansour, W., Hasanain, M., Babulkov, N., Alam, F., Da San Martino, G., Elsayed, T., & Nakov, P. (2021). In G. Faggioli, N. Ferro, A. Joly, M. Maistro, & F. Piroi (Eds.), *CEUR workshop proceedings*, *Overview of the CLEF-2021 CheckThat! Lab Task 2 on detecting previously fact-checked claims in tweets and political debates*. CEUR-WS.org.
Shliselberg, S.-H. M., & Dori-Hacohen, S. (2022). *RIET Lab at CheckThat! 2022: improving decoder based re-ranking for claim matching*: *Working notes of CLEF*.
Shtok, A., Kurland, O., & Carmel, D. (2009). Predicting query performance by query-drift estimation. In *Conference on the theory of information retrieval* (pp. 305–312). Springer.
Shtok, A., Kurland, O., Carmel, D., Raiber, F., & Markovits, G. (2012). Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems (TOIS)*, *30*(2), 1–35.
Song, K., Tan, X., Qin, T., Lu, J., & Liu, T.-Y. (2020). MPNet: Masked and permuted pre-training for language understanding. In *Advances in neural information processing systems, Vol. 33* (pp. 16857–16867). Curran Associates, Inc..
Thorne, J., & Vlachos, A. (2018). Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th international conference on computational linguistics* (pp. 3346–3359).
Thuma, E., Peace, M. N., Tebo, L.-D., & Monkgogi, M. (2020). UB_ET at checkthat! 2020: exploring ad hoc retrieval approaches in verified claims retrieval. In L. Cappellato, C. Eickhoff, N. Ferro, & A. Névéol (Eds.), *CEUR workshop proceedings*, CEUR-WS.org.
Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, *359*(6380), 1146–1151.
Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in neural information processing systems, Vol. 33* (pp. 5776–5788). Curran Associates, Inc..
Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P. N., Ahmed, J., & Overwijk, A. (2020). Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International conference on learning representations*.
Zamani, H., Croft, W. B., & Culpepper, J. S. (2018). Neural query performance prediction using weak supervision from multiple signals. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 105–114).
Zhao, Y., Scholer, F., & Tsegay, Y. (2008). Effective pre-retrieval query performance prediction using similarity and variability evidence. In *European conference on information retrieval* (pp. 52–64). Springer.
Zhou, Y., & Croft, W. B. (2007). Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 543–550).