A Fact Checking and Verification System for FEVEROUS Using a Zero-Shot Learning Approach

Orkun Temiz, Özgün Ozan Kılıç, Arif Ozan Kızıldağ, Tuğba Taşkaya Temizel
Graduate School of Informatics
Middle East Technical University
{orkun.temiz, ozank, arifozan, ttemizel}@metu.edu.tr

Abstract

In this paper, we propose a novel fact checking and verification system to check claims against Wikipedia content. Our system retrieves relevant Wikipedia pages using Anserini, uses BERT-large-cased question answering model to select correct evidence, and verifies claims using XLNET natural language inference model by comparing it with the evidence. Table cell evidence is obtained through looking for entity-matching cell values and TAPAS table question answering model. The pipeline utilizes zero-shot capabilities of existing models and all the models used in the pipeline requires no additional training. Our system got a FEVEROUS score of 0.06 and a label accuracy of 0.39 in FEVEROUS challenge.

1 Introduction

Misinformation on online mediums has caused several problems in recent years. For instance, during the initial spread of the Covid-19 pandemic, inappropriate treatments or incorrect statistics have been widely disseminated through posts. Manually checking the content of such posts against the fact checking sites is not feasible as it is labor intensive. As a remedy, many automated fact-checking solutions have started to emerge in the last decade.

To challenge researchers and advance the domain in this research area, the Fact Extraction and Verification (FEVER) (Thorne et al., 2018) challenge was introduced in 2018. This challenge contained 185,445 claims, and the most successful group (Nie et al., 2019) obtained a 0.63 fever score in the test set. In 2021, a new challenge, Fact Extraction, and VERification Over Unstructured and Structured information (FEVEROUS) (Aly et al., 2021) was organized with a new dataset comprising 87,026 claims where the average length of the claims increased significantly. A Wikipedia dump with more than 5.4 million articles was provided for claim verification, which included sentences and other page elements such as lists and table cells as potential evidence while the previous challenge's dataset contained only sentences. Moreover, the total number of page elements included in the dump increased significantly compared to the previous challenge. Although FEVEROUS challenge contains less number of claims, it has a higher complexity than FEVER challenge. In this challenge, participants were not only required to label each claim as *"SUPPORTS," "REFUTES,"* or *"NOT ENOUGH INFO"* but also provide the correct evidence for it.

The baseline model in FEVEROUS challenge obtained around 18% FEVEROUS score. This model contains two steps, which are retrieval and verdict prediction. The model firstly retrieves relevant pages and then sentences and cells separately from each page. During cell retrieval, tables are linearized to obtain the most relevant cells. Then cell retrieval is handled as a binary sequence labeling task. Verdict prediction is made using the Robustly Optimized BERT Pretraining Approach (RoBERTa) model (Liu et al., 2019). In addition, the FEVEROUS score assumes that the prediction is correct when the label is correct and a set of evidence is present in the predicted evidence.

In this challenge, we developed a pipeline that utilizes zero-shot learning capabilities of existing models where we have considered claims as a question and our retrieved documents as a solution text instead of extracting cells and sentences after the document retrieval. We applied different question answering (QA) models to solve the claim for sentences and cells. We obtained our labels from sentences by using Natural Language Inference (NLI) model. After that, we added the cells after the sentence solutions. Our model obtained 0.06 FEVEROUS score, 0.39 label accuracy, and 0.06 evidence F1 score.

Table 1: The details of the top approaches with respect to document retrieval, sentence retrieval, and claim verification tasks in the first FEVER challenge compared to the FEVER and the FEVEROUS baseline models

	Document Retrieval	Sentence/Cell Retrieval	Claim Verification
FEVER-baseline (Thorne et al., 2018)	TF-IDF	TF-IDF	Decomposable attention
FEVEROUS-baseline (Aly et al., 2021)	TF-IDF	TF-IDF	RoBERTa
UNC-NLP (Nie et al., 2019)	ESIM	ESIM	ESIM
UCL Machine Reading Group (Yoneda et al., 2018)	Logistic regression	Logistic regression	ESIM + aggregation
Team Athene (Hanselowski et al., 2018)	MediaWiki API	ESIM	ESIM

1.1 Related Works

In the first FEVER challenge, the top three groups used Enhanced Sequential Inference Model (ESIM) (Chen et al., 2017) with modifications. The UNC-NLP team (Nie et al., 2019) used Neural Semantic Matching Network (NSMN) for both retrieval and verification tasks while modifying ESIM with additional shortcut connections and changing the output layer to max-pool. Team Athene (Hanselowski et al., 2018) made use of Wikipedia's MediaWiki API to search named entities and ESIM in sentence retrieval and claim verification by extending it to generate a ranking score. This extension adds a hidden layer with a single neuron output and gives the claim together with an input sentence. Finally, UCL Machine Reading Group (Yoneda et al., 2018) employed logistic regression in document and sentence retrieval by utilizing keywords, and features of sentences, respectively. In addition, they aggregated the labels created by ESIM with different models including logistic regression and Multi-Layer Perceptron (MLP) with two layers for claim verification. Their results showed that aggregation with MLP yielded a better result than other aggregation methods.

The FEVEROUS (Aly et al., 2021) baseline model applies TF-IDF for document and sentence retrieval, like the first FEVER challenge (Thorne et al., 2018). On the other hand, the FEVEROUS baseline uses RoBERTa instead of the decomposable attention model for claim verification. Table 1 shows the methodological details of the approaches performed well at the FEVER shared task and the FEVEROUS baseline model. Moreover, Akkalyoncu Yilmaz et al. (2019) retrieved documents while utilizing Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) model for re-ranking results returned by Anserini. Soleimani et al. (2020) applied BERT for sentence retrieval and claim verification while fine-tuning it for each task separately. On the other hand, they made use of Wikipedia's MediaWiki API, similar to Team Athene (Hanselowski et al., 2018) for document retrieval.

2 Method

In the following subsections, we will explain how our system works. An overview of the system in the form of pseudocode is given in Algorithm 1.

2.1 Preprocessing & Keyword Extraction

The claims themselves were directly used to form the base query for retrieving the relevant pages. Since some claims had alternative space characters, these were replaced with a single standard space character. The queries were enhanced with the relevant keywords, which were formed by the named entities extracted from the text using spaCy (n.d.). To improve spaCy's performance, other candidate entities with capitalized first letters were also added. Moreover, to handle the cases where spaCy could not detect the whole word chunk (i.e. Adam Smith), the contingency parser was employed to detect noun chunks. If the named entity found by spaCy was located inside a noun chuck, we added that noun chunk to the keyword list. Date entities

Algorithm 1 The pseudocode of our proposed pipeline

Input:
$claims$: A list including the claims that will be verified.
raw_docs: A database including the Wikipedia documents provided by FEVER.
$indexed\_docs$: A formatted sentence corpus of the provided Wikipedia documents, indexed using Anserini.
Initialize:
$results \leftarrow$ [] // Create a list with eventual dimensions of [$claims.length$,3] to store the claim, predicted label, and predicted evidence
for each c in $claims$ **do**
    Extract entities from claim c using spaCy, uppercase detection, and chunking, store it in $entities$
    Obtain a query by appending $entities$ to claim c, and set it into $query$
    Obtain the most relevant documents based on $query$ from $indexed\_docs$ and set it into $docs[doc\_title, doc\_content, relevance]$
    Apply string matching to document titles from $docs$ with $entities$, maximize relevance scores of the matched documents (see Section 3.1 for its explanation)
    $sentence\_evidence \leftarrow \{\}$ // Create empty sets that will be filled with evidence
    $table\_evidence \leftarrow \{\}$
    for each d in $docs$ **do**
        if $sentence\_evidence.length$ < 5 **then**
            Divide document d into subdocuments $subdocs$ (See Section 2.3)
            for each sd in $subdocs$ **do**
                Retrieve the evidence sentence and its immediate predecessor as ev using c, sd, and BERT QA model
                Add $ev.sentence\_ids$, doc_title of d, and ev's confidence score (according to the universal sentence encoder) to $sentence\_evidence$
            end for
        end if
        if $table\_evidence.length$ < 25 **then**
            Extract the document tables from raw_docs, normalize their formats (see Section 2.3), and set them into $tables$
            Retrieve cell values from $tables$ that match $entities$, and add all the cell IDs from their corresponding rows to $table\_evidence$
            Apply TAPAS QA model to c and $tables$, and add the retrieved cells' IDs to $table\_evidence$
        end if
    end for
    Rank evidence in $sentence\_evidence$ based on confidence scores
    // Ensure evidence do not exceed the limit (5 and 25 for sentence and table evidence, respectively) by slicing them, combine them and push to the results list
    $predicted\_evidence \leftarrow sentence\_evidence$[0:5] + $table\_evidence$[0:25]
    Verify claim c using c, $sentence\_evidence$[0:5], and XLNET model, set it into $predicted\_label$
    $results$.push([c, $predicted\_label$, $predicted\_evidence$])
end for
Output:
$results$: A list with claims, their predicted label, and predicted evidence.

were ignored. These obtained entities were concatenated twice to the query to give them more weight, since our document retrieval module uses OKAPI BM25 (Robertson et al., 1994), where including a phrase more than once causes the module to give it more weight in the document retrieval process.

2.2 Document Retrieval

To retrieve Wikipedia texts efficiently, we use Anserini indexing, which uses OKAPI BM25 (Robertson et al., 1994) for indexing the Wikipedia pages. Anserini (Yang et al., 2017) is a toolkit developed on Apache Lucene, open-source search software. To use Anserini indexing, we transformed the Wikipedia dump into an indexable format while discarding lists, tables, and section titles. Then, we indexed it using Anserini toolkit with the help of Pyserini (Lin et al., 2021), a Python interface for Anserini. We fed the query and the keywords in a concatenated way to the Anserini. By retrieving 70 pages per claim, and also obtaining the documents that link to the retrieved relevant documents, the algorithm could successfully retrieve all of the documents that have the necessary evidence for 7255 claims out of the 7891 (91.94%) from the development set. However, we later saw that better document retrieval does not always translate well to evidence retrieval and verification. These settings were causing the retrieved evidence to be noisy and taking too much time. Therefore, looking for the incoming links was later scrapped, and only 10 documents were retrieved for every claim to speed up the process.

2.3 Evidence Selection

In this section, we selected the sentences related to or considered as potential evidence with respect to the query from the retrieved Wikipedia pages. To select the relevant parts, we employed BERT-large-cased (Devlin et al., 2019) question answering model instead of a sentence similarity model even though the claims were not including a question. Although sentence similarity models were highly used in FEVER (Thorne et al., 2018) tasks, with the help of QA models, search may grasp the nuance and semantic meaning of the query better than sentence similarity models. In line with our approach, Google also employs a BERT question answering model for its searches (Nayak, 2019).

Since BERT is able to handle a maximum of 512 tokens at once and Wikipedia pages contain long texts, we split the retrieved text into chunks of 10 sentences. This way, we were also able to retrieve more than one answer from one document, since we would get an answer from each split. Although splitting the page helped with the token limit, it did not ensure that truncation would not occur. For this purpose, after the initial split, we split the chunks further with 10% overlapping words into chunks of at most 512 words in order not to lose the semantic meaning of the chunks. Then, the QA model was applied to all the chunks and the answer with the highest score was labeled as the final answer. As evidence identifiers such as "sentence_1" created noise and negatively affected the QA model, these identifiers were cleaned. After that, we retrieved the sentence including the answer and its preceding one to ensure to obtain the full answer. Correct evidence identifiers were then obtained through the returned pieces of evidence as the answer. Then, we sorted them with the universal sentence encoder according to its similarity (confidence) score with the query (Cer et al., 2018). We found that retrieving pages related to people with very similar names to the "PERSON" entity in the query was throwing the results off. To tackle this issue while sorting the answers, we doubled the similarity score obtained after a softmax normalization if the document title matched the "PERSON" named entity recognized by the spaCy. Page title and person entity are considered to be matched when one includes the other. For the task scoring constraints, we kept only the top 5 pieces of evidence. As a result of the textual QA module, we ended up with a query, its answers and the confidence scores between the query and the textual answers. Also, note that since we got a sentence which included the answer, and the sentence before it, a full answer text may contain more than one evidence. In Wikipedia pages, evidence or answers may have been located in the provided table cells. To address this gap, we employed two methods; The first method involved using the non-person entities from the claim and matching them with the cell values from the tables of the relevant page. A cell value was considered to be a match when its original or link-removed version had a Levenshtein ratio of 0.8 or higher with the non-person entity.

The second method involved using TAPAS (TAble PArSing) (Herzig et al., 2020), a weakly supervised transformer-based question answering model developed by Google Research. Given a table with column header names and cell values, the model

can predict the answer according to the given query, similar to the textual question answering model, except our answer is cell values instead of text chunks. To make this method work, tables of relevant pages were obtained in a normalized form such that cells with row/column spans larger than one are divided into 1x1 cells sharing the same value, compatible with the model. Since TAPAS requires tables to have column headers, the table rows were removed from the beginning one by one until the first row included the header cells. Dividing the cells into 1x1 cells and duplicating their values led some tables to be very crowded and caused memory issues. To address this problem, firstly, if a row has more than 700 characters combined, the row is removed. Secondly, if the final table has more than 1000 tokens when it is tokenized, it is skipped as a whole. We chose to do so due to the time constraints and our anecdotal findings of marginally large tables having tangential information.

These two methods were applied to the pages in order from the highest to the lowest confidence scores, and only the first 25 cells (belonging to the most relevant evidence's pages) were kept. Since cells usually do not form complete sentences, we did not use them in the textual entailment step to decide whether a claim is supported or not. Internal links to other Wikipedia pages are formatted in the dataset as "[[Page_ID|Visible text]]" where "Page_ID" denotes the identifier through which the page can be accessed (like "https://en.wikipedia.org/wiki/Page_ID") while "Visible text" denotes the text (mostly the linked page's title) shown to the user. Since these links create noise and prevent matches, they are simplified to obtain plain text cells with both cell evidence retrieval methods.

2.4 Textual Entailment

For the entailment model, we used XLNET (Yang et al., 2019) trained on the composition of SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), FEVER (Thorne et al., 2018), ANLI (Williams et al., 2020) and NLI (Nie et al., 2019) datasets. By using the pre-trained model, we evaluated the entailment between the textual answers and the query. As a result of the textual entailment model, we retrieved the Support, Contradict or Neutral (*NOT ENOUGH INFO*) scores between one query and one answer instance.

2.5 Heuristic Verdict Assignment

After utilizing the textual entailment module, we concluded the final verdict, which will be one of Support, Contradict or Neutral (*NOT ENOUGH INFO*) via the following heuristic:

- If there was no answer with a similarity score of 0.6 between the query and the answer threshold, it was assigned as Neutral.
- If the "neutral" score between the query and the answer was higher than 0.8, it was counted as a Neutral vote. For the other cases, we look for the contradiction and entailment scores between the query and the answer. If the entailment score was higher, we added one vote for Support label. If the contradiction score was higher, we added one vote for the Contradict label.
- At the end, a majority vote was taken between the "Support" and "Contradict" label votes and then we determined the final verdict.

If the outcome is Neutral, we can conclude that there is no information, which support or contradict the claim in the Wikipedia pages. Even if the verdict was *NOT ENOUGH INFO*, we still fetched the evidence as well.

3 Results and Discussion

Based on the official leaderboard (Fact Extraction and VERification, 2021), our pipeline's scores along with the baseline, minimum, and maximum FEVEROUS scores are shown in Table 2. Excluding the baseline, our FEVEROUS score is the ninth out of 12 groups.

The label accuracy of our method (0.39) is relatively close to the baseline (0.48). For reference, the accuracy obtained with random guesses on the development set is 0.33 while randomly guessing the label using the class distribution yields an accuracy of 0.45.

Our pipeline's success in identifying the expected evidence and consequently our FEVEROUS score are significantly lower than the baseline. Having a recall of 0.10 and precision of 0.05 suggests we have more false positives than false negatives. The fact that we also retrieved the previous sentence of the sentence retrieved from the question answering model may have an effect on this, but a significantly lower evidence precision compared to

Table 2: Our model's results compared to the baseline, minimum, and maximum FEVEROUS scores among the participant groups

	FEVEROUS Score	Accuracy	Evidence F1	Evidence Precision	Evidence Recall
Maximum	0.2701	0.5607	0.1308	0.0773	0.4258
FEVEROUS Baseline	0.1770	0.4760	0.1610	0.1121	0.2855
Ours	**0.0636**	**0.3897**	**0.0634**	**0.0462**	**0.1011**
Minimum	0.0223	0.3999	0.0282	0.0245	0.0330

the evidence recall is seemingly the norm among the participant groups.

We had not run the final version of the pipeline on the whole development set before our test submission. We later ran it on the development set and obtained very similar results (a FEVEROUS score of 0.0642 and label accuracy of 0.3867), which suggests dataset splits are well-balanced. The confusion matrix for the development set is shown in Table 3.

Table 3: Confusion matrix for the development set predictions with the base model, *"N.E.I."* indicating *NOT ENOUGH INFO* label

		Predicted		
		SUPPORTS	REFUTES	N.E.I.
Actual	SUPPORTS	1009	1366	1533
	REFUTES	530	1834	1117
	N.E.I.	112	181	208

3.1 Limitations, Improvements, and Future Work

Numerous improvements can be made on the pipeline. Based on the confusion matrix for the development set, our pipeline is seemingly too much inclined towards finding that there is not enough information as 2858 claims from the development set are labeled as *NOT ENOUGH INFO* compared to the expected number of 501 while only 208 of them were true positives. This suggests the claim verification model requires fine-tuning. Even a more heuristic solution can slightly improve the results. Since the mean and median number of expected non-cell evidence were approximately two, we observed that randomly assigning the label as either *SUPPORTS* or *REFUTES* for claims that have more than two retrieved non-cell evidence increases the label accuracy to 0.51 and the FEVEROUS score to 0.07. We found that it is also possible, to some extent, to verify the claims based on whether the retrieved documents mention entities extracted from the claim. We naively assumed that if a claim's entities are completely matched in the documents, the claim is correct. If most of the entities were not found, then there was not sufficient information. Using the ratio of entities matched in the documents and some threshold values, we obtained a lowered FEVEROUS score (0.0457) but a higher label accuracy (0.4593). More importantly, we saw that applying this naive approach only when the predicted label is wrong (4839 claims out of 7890) significantly improves both FEVEROUS score (0.0812) and label accuracy (0.6718). While this is not applicable when we do not have the expected label, this suggests that a complementary naive approach can significantly improve the results if we can identify which cases are more likely to be misclassified.

Due to time constraints, several parameters (like the number of retrieved documents) were kept at minimal and not optimized. While we limited document retrieval to 10 results per claim, we had observed that some expected Wikipedia pages for the claims in the development set were being retrieved at much lower ranks (such as 50 and up). As explained, deep learning models used within the pipeline are pre-trained models that are not fine-tuned for this task. Fine-tuning these parameters and models may yield better results. However, using a subset of the development set, we saw that retrieving 70 results for each claim only improved the label accuracy by 0.02 and FEVEROUS score did not change while the pipeline became 5-6 times slower.

We found that people's names in the claims and their Wikipedia pages do not always perfectly match. For example, Sir Arthur Conan Doyle was mentioned in a claim as *"Conan Doyle"* while his Wikipedia page was called *"Arthur Conan Doyle."* Our pipeline requires the entity to be included in

the page title or the page title to be included in an entity to double its confidence score, so these simple differences could be handled. However, while it is rarer, we saw that certain entities have more significant differences between their mentioned names and their Wikipedia page titles. For example, Eleanor Francis *"Glo"* Helin was mentioned as *"E. F. Helin"* in a claim while her Wikipedia page was titled *"Eleanor F. Helin."* For these cases, removing the disambiguation parentheses, using a Levenshtein ratio threshold, and initial matching when there is initialism involved may improve the results. Since these name differences can be seen anywhere, a more flexible and tolerating approach may be helpful while dealing with entities. With a subset of the development set, using Levenshtein ratio as an alternative to partial matching (without dealing with initialism) increased FEVEROUS score by about 0.01, which is not significant.

Fundamentally changing some parts of the pipeline may have positive effects as well. Our question answering model can retrieve multiple evidence sentences, but the retrieved sentences must be in consecutive order since the model actually retrieves a piece of the document that is deemed relevant which can span multiple sentences. This limitation is partially alleviated since we split all the documents into chunks of sentences before using the question answering model. Separately feeding each piece of evidence or splitting the documents using a sliding window approach with a small window may improve the results, but it would also increase the inference time. Similarly, splitting the tables further can prevent truncation and may improve evidence recall. While tables may mislead the model, the existence of matching entities in them may, in general, give some clues about the claim's veracity. Based on the complicatedness of the claims, it might be possible to improve the results by adjusting the entailment score when there is table cell evidence.

Inference takes a considerable amount of time as the pipeline becomes more complex with multiple models. Due to verification and evidence retrieval taking roughly eight or nine seconds per claim, we ran the pipeline with two computers in parallel. Reducing the inference time can help with speeding up the iterative improvements and experimentation.

4 Conclusion

In this work, we proposed a fact extraction and verification pipeline that mainly uses Anserini to retrieve documents, a BERT-based question answering model to retrieve textual evidence, TAPAS to retrieve table cell evidence, and an XLNET-based entailment model to judge the claim without fine-tuning them. We believe parameter optimization and challenge-specific fine-tuning can significantly improve the results.

References

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with birch. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 19–24, Hong Kong, China. Association for Computational Linguistics.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: Fact Extraction and VERification over unstructured and structured information.

Apache Lucene. Apache Lucene - welcome to Apache Lucene.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fact Extraction and VERification. 2021. 2021 shared task. [Online] Available at: https://fever.ai/task.html [Accessed August 8, 2021].

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-Athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2356–2362, New York, NY, USA. Association for Computing Machinery.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pandu Nayak. 2019. Understanding searches better than ever before. [Online] Available at: https://blog.google/products/search/search-language-understanding-bert/ [Accessed August 8, 2021].

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1994. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. BERT for evidence retrieval and claim verification. In *Advances in Information Retrieval*, pages 359–366, Cham. Springer International Publishing.

spaCy. n.d. spaCy · Industrial-strength Natural Language Processing in Python. [Online] Available at: https://spacy.io/ [Accessed August 8, 2021].

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Adina Williams, Tristan Thrush, and Douwe Kiela. 2020. ANLIzing the adversarial natural language inference dataset. *arXiv preprint arXiv:2010.12729*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL machine reading group: Four factor framework for fact finding (HexaF). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102, Brussels, Belgium. Association for Computational Linguistics.