Language Models as Fact Checkers?

Nayeon Lee[1*] **Belinda Z. Li**[2] **Sinong Wang**[2]
Wen-tau Yih[2] **Hao Ma**[2] **Madian Khabsa**[2]
[1]Hong Kong University of Science and Technology [2]Facebook AI
nayeon.lee@connect.ust.hk
{belindali,sinongwang,scottyih,haom,mkhabsa}@fb.com

Abstract

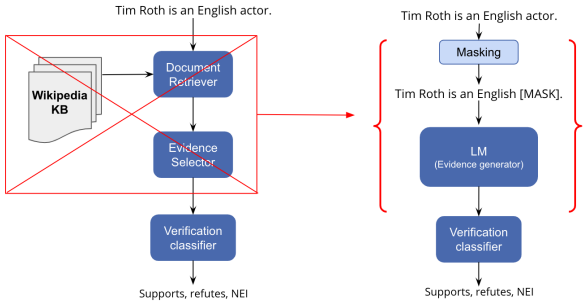
Recent work has suggested that language models (LMs) store both common-sense and factual knowledge learned from pre-training data. In this paper, we leverage this implicit knowledge to create an effective end-to-end fact checker using a *solely* a language model, without any external knowledge or explicit retrieval components. While previous work on extracting knowledge from LMs have focused on the task of open-domain question answering, to the best of our knowledge, this is the first work to examine the use of language models as *fact checkers*. In a closed-book setting, we show that our zero-shot LM approach outperforms a random baseline on the standard FEVER task, and that our finetuned LM compares favorably with standard baselines. Though we do not ultimately outperform methods which use explicit knowledge bases, we believe our exploration shows that this method is viable and has much room for exploration.

1 Introduction

Pre-trained language models have recently lead to significant advancements in wide variety of NLP tasks, including question-answering, commonsense reasoning, and semantic relatedness (Devlin et al., 2018; Radford et al., 2019; Peters et al., 2018; Radford et al., 2018). These models are typically trained on documents mined from Wikipedia (among other websites). Recently, a number of works have found that LMs store a surprising amount of world knowledge, focusing particularly on the task of open-domain question answering (Petroni et al., 2019; Roberts et al., 2020). In this paper, we explore whether we can leverage the knowledge in LMs for *fact checking*.

We propose an approach (Fig. 1b) that replaces the document retriever and evidence selector models in traditional fact-checking (Fig. 1a) with a single language model that generates masked tokens. This offers a number of advantages over the traditional approach: first, the procedure is overall simpler, requiring fewer resources and computation – we do not need to maintain an explicit knowledge base external to our LM, and we do not need an explicit retrieval step. The latter in particular can lead to a huge speedup in the system, since we can skip the time-consuming step of searching over a potentially massive space of documents. Second, LMs are widely-available and are currently attracting significant research effort. Thus, research in language-modeling, particularly in improving LMs ability to memorizing knowledge, may also improve the overall effectiveness of our fact-checking pipeline. Lastly, our system further shifts the paradigm towards "one model for all" — LMs have been used for a wide variety of tasks, and now also for fact checking.

In order to determine the feasibility of our approach, we start with a human review study where participants are given a claim from FEVER (Thorne et al., 2018a), and are asked to validate the claim using only a BERT language model. We found that users had reasonable success in determining claim validity. Empowered by the results,



(a) Traditional fact-checking pipeline.

(b) Our new fact-checking pipeline.

Figure 1: Traditional fact-checking pipeline (left) vs. Our LM-based pipeline (right)

*Work done while at Facebook AI.

we design an *end-to-end* neural approach for utilizing BERT as a fact checker (see Figure 1b). At a high level, we first generate an evidence sentence by masking the claim and using BERT to "fill in" the mask. We then feed the generated sentence, alongside the original claim, to a verification classifier model that classifies whether the claim is supported, refuted, or the information is insufficient to make a call.

The rest of the paper is organized as such: Section 2 gives an overview of the problem space. Section 3 describes our preliminary experiments. Sections 4 and 5 highlights our main methods (i.e. end-to-end model, experimental setup), and 6 reports our main results. Sections 7 and 8 conclude our paper with a discussion and future works.

2 Background

Task The main goal of fact-checking is to validate the truthfulness of a given claim. Each claim is assigned one of three labels: support, refute, or not enough information (NEI) to verify.

Dataset We use FEVER (Thorne et al., 2018a), a large-scale fact-checking dataset with around 5.4M Wikipedia documents. Claims were generated by extracting sentences from Wikipedia (with possible mutations), and were annotated by humans with their verification label and/or evidence sentences from Wikipedia.

Traditional pipeline Traditional fact-checking systems (Fig. 1a) access knowledge within an external knowledge base (i.e. Wikipedia) to validate a claim. They use a multi-step, pipelined approach, which involve IR-modules, such as document retrievers and evidence selectors, for retrieving the appropriate evidence, and verification modules that take in {`claim, [evidences]`} pairs and predict a final verification label

Our pipeline As shown in Fig.1b, our proposed pipeline replaces both the external knowledge base as well as the IR modules with a pretrained language model. In the remainder of this paper, we utilize BERT. Future work can explore other language models.

Querying the Language Model In Petroni et al. (2019), language models were used as knowledge base to answer open-domain questions. To do this, the authors devised a probe known as "LAMA", which generates fill-in-the-blank cloze-style statements from questions. For example, in order to answer the question 'Where is Microsoft's headquarter?', the question would be rewritten as as 'Microsoft's headquarter is in [MASK]' and fed into a language model for the answer.

Inspired by LAMA (Petroni et al., 2019), we also generate evidences from language models through fill-in-the-blank style tasks.

3 Exploratory Experiments

In order to determine the feasibility of our approach, we began by conducting a human review study on 50 random-selected claims from FEVER (Thorne et al., 2018a). Participants were asked to validate each claim with *only* a language model, by following these steps:

1. Mask a token from the claim, depending on component of the claim we wish to verify:
   Thomas Jefferson founded the University of Virginia after retiring → Thomas Jefferson founded the University of [MASK] after retiring.
   In this example, the user is verifying which university was founded by Thomas Jefferson. Note that the user could alternatively choose to mask *Thomas Jefferson* in order to verify the founder of University of Virginia.

2. Get the top-1 predicted token from the LM.
   Top-1 predicted token = Virginia.

3. If predicted token matches the masked token, the claim is supported, otherwise it is refuted.
   Virginia ≡ Virginia → SUPPORTS

In other words, we asked participants to serve as the "masking" and "verification classifier" components of our fact-checking pipeline in Fig. 1b.

Two participants examined the 50 claims, and eventually achieved an average accuracy of 55%. [1]

We also conducted this zero-shot study on a larger scale and in a more systematic way, by taking all claims in the *full* FEVER dataset, and always masking the last token.[2] Otherwise, we preserve steps 2 and 3 from above. Even with this naïve

[1] Both participants had NLP background, and both were familiar with FEVER and the fact-checking task. We also assumed both participants were capable of selecting the optimal position to mask.

[2] We omit examples for which the masked token is not in BERT's vocab.

token-matching approach, we were able to obtain precision 56% and F1 59% for the positive label (SUPPORT).

Our preliminary experiments' results illustrate that, with a good masking mechanism and verification model, language models can indeed feasibly be used for fact-checking.

4 End-to-End Fact-Checking Model

Enlightened by results from our preliminary experiments, we devise an end-to-end model that automates and improve upon the masking and verification steps that were conducted by humans. Specifically, we resolve two limitations: 1. manual masking of claims, and 2. naïve validation of the predicted token that fails to deal with synonyms and other semantic variants of the answer.

Automatic Masking We mask the *last named entity* in the claim, which we identify using an off-the-shelf Named-Entity-Recognition (NER) model from spaCy Honnibal and Montani (2017). In particular, we choose to mask named entities in order to better ensure that the token we mask actually makes use of the *knowledge* encoded in language models. (Otherwise, we may mask tokens that only make use of the LM's ability to recover linguistic structures and syntax – for instance, masking stopwords). This hinges on the observation that, for most claims, its factuality hinges upon the correctness of its entities (and the possible relations between them), and *not* on how specifically the claim is phrased.

Verification using Entailment To move beyond naïvely matching predicted and gold tokens, we leverage a textual entailment model from AllenNLP (Gardner et al., 2018) to validate our LM predictions. Note that textual entailment models predict the directional truth relation between a text pair (i.e. "sentence t entails h" if, typically, a human reading t would infer that h is most likely true).

Full-pipeline steps Detailed steps for our end-to-end model (Fig. 2) are as follows:

1. Masked the last named entity found by the NER model.

2. Get the top-1 predicted token from the LM, and fill in the [MASK] accordingly to create the "evidence" sentence.

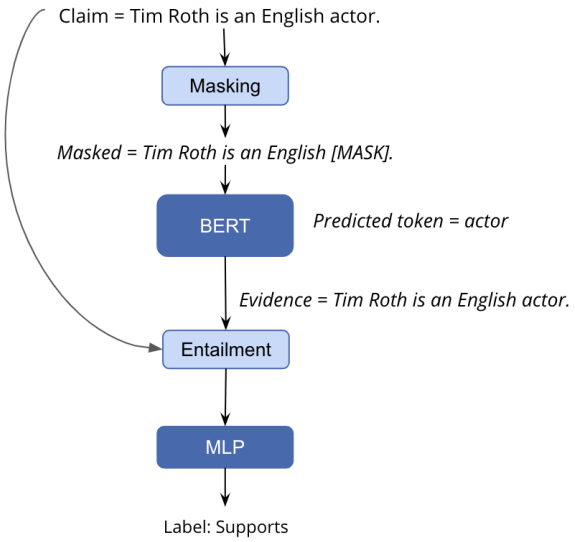


Figure 2: Detailed illustration of our pipeline

3. Using the claim and generated "evidence" sentence, obtain entailment "features" using outputs from the last layer of the pretrained entailment model (before the softmax).

4. Input the entailment features into a multi-layer perceptron (MLP) for final fact-verification prediction.

5 Experiments

5.1 Experiment setup

We conduct our experiments on the FEVER claim verification dataset (Thorne et al., 2018a) using the standard provided splits. We use the publicly available 24-layer BERT-Large as our language model, which was pre-trained on Wikipedia in 2018.[3]

The MLP was optimized using Adam, and trained with a mini-batch size of 32. The learning rate was set to 0.001 with max epoch size 200 and epoch patience of 30. The embedding size of the entailment features (from the pre-trained entailment model) was 400, and our MLP classifier had hidden size of 100.

5.2 Evaluation Metric

The traditional pipeline was evaluated using FEVER scoring, which is a stricter form of scoring that treats predictions to be correct only when correct evidences were retrieved. Since our pipeline

[3]It's possible the model was trained on a later Wikipedia dump than what's released as part of FEVER, but pre-training BERT from scratch is beyond the scope of this paper.

Model	Label	prec	recall	f1	accuracy	macro prec	macro recall	macro f1
$BERT_{freeze}$	REFUTES	0.36	0.69	0.47	0.38	0.39	0.38	0.33
	SUPPORTS	0.43	0.09	0.15				
	NEI	0.39	0.35	0.37				
$BERT_{finetune}$	REFUTES	0.62	0.55	0.58	0.57	0.57	0.57	0.57
	SUPPORTS	0.54	0.67	0.59				
	NEI	0.57	0.49	0.53				
$BERTasKB$	REFUTES	0.76	0.38	0.51	0.49	0.59	0.49	0.44
	SUPPORTS	0.41	0.92	0.57				
	NEI	0.58	0.15	0.24				
Shared-Task Baseline (Thorne et al., 2018b) *	-	-	-	-	**0.49**	-	-	-
Shared-Task SoTA (Thorne et al., 2018b) *	-	-	-	-	**0.68**	-	-	-
DREAM SoTA (Zhong et al., 2020) *	-	-	-	-	**0.77**	-	-	-

Table 1: Performance comparison between BERT-as-encoder models ($BERT_{freeze}$, $BERT_{finetune}$) and BERT-as-LM model ($BERTasKB$). New SoTA result (DREAM) released in ACL2020 is added for reference (*We report fact-checking label accuracy, not the stricter FEVER score)

does not utilize an external knowledge base, and does not have an evidence retriever, we only examine the correctness of the final verification step using precision, recall, F1 and accuracy. We leave generating evidences with language models for future work.

5.3 Baselines

We introduce two language model baselines for comparison. The first baseline, $BERT_{freeze}$, uses an MLP layer on top of a frozen BERT encoder to make predictions (gradients backpropagate to the MLP layer only). In this baseline, we aim to *extract* the already stored knowledge within BERT model as an embedding vector, and avoid finetuning the internal layers, in order to disentangle BERT's knowledge from it's ability to serve as a high-capacity classifier.

The second baseline, $BERT_{finetune}$, allows all the model layers to be updated based on the fact-verification loss from the MLP layer. This baseline captures BERT's ability as *both* a language model, and a high-capacity text encoder.

Note that the dataset is evenly distributed among the three classes, therefore a random baseline would yield an accuracy of 33%. Also note that the Fever-baseline model introduced by the task organizers achieves accuracy score of 48.8% (Thorne et al., 2018b).

6 Results and Discussion

The results of the three models are reported in Table 1. We observe that our proposed approach ($BERTasKB$) outperforms $BERT_{freeze}$ on all metrics suggesting that querying language models in QA style is a better approach for extracting their encoded knowledge. Similarly, $BERTasKB$ model achieves an accuracy score of 49% which is comparable to Fever-baseline at 48.8%, except without the need for explicit document retrieval and evidence selection. This suggests that language models, used as sources of knowledge for fact checking, are at least as effective as standard baselines. However, there is still much room for future research, as the state-of-the-art model on the Fever shared task achieves an accuracy score of 68.21% (Thorne et al., 2018b).

On the other hand, we find that $BERTasKB$ lags behind $BERT_{finetune}$, as expected, on most metrics. We hypothesize this is due to the high capacity of the model, in comparison, and to the effectiveness of BERT models in text classification. Upon examining the results of these two models closely, we find that $BERTasKB$ struggles mightily with the `NEI` category (F1 score of 0.24 vs 0.53) indicating that our current approach might need specific modules to better tackle that category. As both models seem to be equally adept in identifying the `support` class (0.57 vs 0.59 F1), indicating that $BERTasKB$ is unable to distinguish between `refute` and `NEI` classes. Future work can further investigate techniques to identify these two categories.

Interestingly, the $BERT_{freeze}$ achieves an accuracy score of 38% which is slightly better than a random baseline which achieves 33%.

7 Analysis of Token Prediction Results

In this section, we provide some examples of tokens predicted from BERT to understand the performance of "evidence generation".

First two examples in Table 2 (a, b) are exam-

ID	Claim	Masked Token	Predicted Token	Label
a	Kuching is the capital of [MASK].	Sarawak	Sarawak	SUPPORTS
b	The Beach's director was Danny [MASK].	Boyle	Boyle	SUPPORTS
c	Tim Roth was born in [MASK]	1961	London	SUPPORTS
d	Chile is a [MASK].	country	democracy	SUPPORTS
e	Seohyun [MASK].	sings	Park	SUPPORTS

Table 2: Examples of token predictions from BERT in zeroshot setting. a, b are correctly fact-checked examples, and c, d, f are wrongly fact-checked examples.

ples with correct fact-check labels from zeroshot setting. When a claim has enough context, and contains rather rare names such as "Sarawak", BERT manages to predict correct tokens.

We also provide detailed analysis on the error cases to facilitate future work in making further improvements:

- One common form of errors is that, the entity type of token prediction is biased towards the way how the training data was written. For example, sentence c from Table 2 illustrates a common claim structure in FEVER dataset which talks about the birth-year of a person (e.g., Tim Roth). However, 100% of our test samples with such structure always predict city/country (e.g., London). The reason is, in Wikipedia, the birth-years are always written in the following structure "PERSON (born DATE)" (e.g., "Tim Roth (born 14 May 1961)"), and birth city/country written in "PERSON was born in city/country" structure (e.g., "Roth was born in Dulwich, London"). Therefore, to obtain birth-year, the claim had to be written as Tim Roth (born [MASK]) to predict correctly.

- Sentence d is another example that the entity type of token prediction is hard to control. "is a..." is a very general prefix phrase, making it hard for BERT model to correctly predict correct entity type.

- There are lots of short claims in FEVER test set (approx. 1100 samples) which has less than 5 tokens (e.g. sentence e). Since there is very little context, BERT struggles to predict correctly.

One of the the main insight we get from these analysis is that, the way the language model is initially pre-trained, greatly determines the way it should be "queried".

8 Conclusions & Future Work

In this paper, we explored a new fact-checking pipeline that use language models as knowledge bases. Unlike previous pipelines that required dedicated components for document retrieval and sentence scoring, our approach simply translates a given claim into a fill-in-the-blank type query and relies on a BERT language model to generate the "evidence". Our experiment shows that this approach is comparable to the standard baselines on the FEVER dataset, though not enough to beat the state-of-the-art using the traditional pipeline. However, we believe our approach has strong potential for improvement, and future work can explore using stronger models for generating evidences, or improving the way how we mask claims.

In the future, we will investigate sequence-to-sequence language models such as BART (Lewis et al., 2019) or T5 (Raffel et al., 2019), that have recently shown to be effective on generative question-answering (Roberts et al., 2020). Similarly, our proposed approach seem to struggle with correctly identifying NEI cases, and we plan to investigate adding specific modules to deal with NEI. Lastly, we plan to explore new ways of pre-training language models to better store and encode knowledge.

Acknowledgements

We would like to thank Fabio Petroni for the helpful discussion and inspiration.

References

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

bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and verification (fever) shared task. *arXiv preprint arXiv:1811.10971*.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. *arXiv preprint arXiv:1909.03745*.