Extracting, and not extracting, knowledge from language models for fact-checking

Stanford CS224N Custom Project

Charlie Curnin
Department of Computer Science
Stanford University
ccurnin@stanford.edu

Abstract

Fact-checking is a challenging and useful classification task where a model evaluates the truthfulness of a natural-language claim. Researchers have taken a variety of approaches to building automated fact-checking systems, but recent work has introduced a paradigm that queries a language model to extract factual knowledge. We implement and evaluate several extensions to that pipeline, including alternate strategies for masking the claim and selecting the tokens that the language models predicts for masks. Evaluating these alternatives on a well-known fact-checking dataset, we find that they find that they have minimal impact on overall performance. Motivated by this finding, we construct a drastically simplified version of the pipeline — removing the language model — and find that its accuracy changes little. While its performance remains below state-of-art, these surprising results highlight difficulties in extracting knowledge from language models and introduce a new (to our knowledge) kind of entailment-based fact-checking baseline that involves no language model, corpus or knowledge base.

1 Key Information to include

- Mentor: Megan Leszczynski
- External collaborators: None
- Sharing project: None

2 Introduction

Effective automated fact-checking will have applications aplenty. Reliable assessments of the truthfulness of natural-language claims can save journalists time, debunk spurious claims online and rein in disinformation proliferating on social media.

Fact-checking is difficult, though, even for trained human experts. A claim's truthfulness depends not just on its wording, but also its context and the state of the world. Evaluation of a claim — which may take the form of a two-way classification over true and false, or a gradated assessment of truthfulness — requires factual knowledge, not just linguistic understanding.

This is part of what makes developing automated systems for fact-checking so interesting. Fact-checking implicates a range of natural-language tasks, from coreference resolution to sarcasm detection, but knowledge is central. The task is a testbed for assessing techniques to embed knowledge in, and extract knowledge from, computational models.

Recent work has focused attention on a new method for building automated fact-checkers: leveraging language models and their capacity to hold factual knowledge (LMs). Lee et al.'s fact-checking paradigm queries an LM with a masked version of a claim, and makes a classification by using a textual entailment system and neural net to evaluate how the LM's prediction relates to the claim [1].

We explore several techniques for extending that approach, including different ways of masking an input claim and selecting tokens from an LM. We hoped these alternatives would improve performance by extracting more information from the LM and addressing challenges with extracting knowledge from LMs. Instead, we found that they have minimal impact on the fact-checking system's accuracy.

Motivated by this finding, we prepare a drastically simplified version of the pipeline, removing the LM altogether. A claim travels directly to an off-the-shelf textual entailment system as both premise and hypothesis, and a neural net makes a classification based on features from the entailment model. Remarkably, this system, although it cannot draw knowledge from an LM, corpus or knowledge base, has comparable accuracy to the original pipeline.

3 Related Work

Approaches to fact-checking.

Exact formulations of fact-checking as a task differ, and so do approaches to that task. Researchers have modeled fact-checking as a three-way [2], four-way [3] and five-way [4] classification task, allowing for gradations in truthfulness and the possibility that a given textual source may not speak to a given claim.

To build fact-checking systems, researchers, exploiting connecting between fact-checking and other tasks, have applied methods including textual entailment, relation extraction and textual similarity methods, sometimes with the help of a knowledge base or external corpus [5]. Another paradigm is for systems to connect modules with distinct functions, which might include finding relevant documents, extracting relevant passages from documents and using passages to make a classification about a claim [3] [2].

Boosted by the availability of fact-checking datasets like FEVER [2], research on the task has grown.

Language models in fact-checking

One new kind of fact-checking approach relies on language models, which have been shown to capture knowledge not just about language but also the world [6]. Generally speaking, the strategy is to mask a token in a given claim, have an LM predict what the mask hides and then make a classification based on the relationship between the original input and the LM's prediction.

One system masks named entities in FEVER claims, and makes a classification by applying a threshold to the ratio of these pseudo-questions where the LM prediction aligned with the input [7]. They report accuracy of 80%, but reduce the task from a three- to a two-way classification over `SUPPORTS` and `MANUAL_REVIEW`. They also fine-tune the LM.

Lee et al.'s system is similar but attempts a harder problem, retaining the task as three-way classification and refraining, in their primary pipeline, from fine-tuning the LM [1]. They mask just the last named entity in a claim, and use an LM to predict what the mask hides. The original claim, and the sentence filled by the LM, are input to a pretrained textual entailment model. Finally, a feedforward neural network predicts, based on features from the entailment model, whether the claim is supported, refuted or there's not enough information.

The proof-of-concept approach has lower accuracy than state-of-art methods, and cannot provide "evidence" for its judgments like some other systems. But it performs comparably to standard baselines, doesn't require a corpus or knowledge base and seems to tie fact-checking performance to the renown capabilities of LMs. We seek to extend this approach.

4 Approach

We take Lee et al.'s approach as a baseline. We explore alternative strategies for masking a claim and selecting tokens from the LM. And we prepare a reduced version of the pipeline that removes the LM altogether.

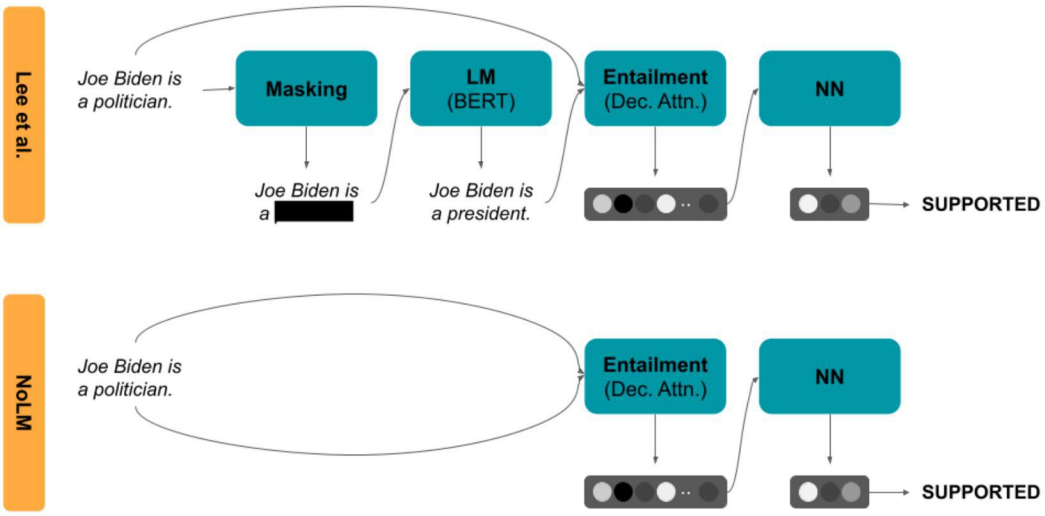

Figure 1: The language-modeling approach to fact-checking from Lee et al., and an alternative with the language model and masking removed.

4.1 Baseline

Given a claim to evaluate, Lee et al.'s pipeline masks the last named entity in the claim, and has the language model BERT [8] predict what the mask hides. The resulting text, as well as the initial claim, are input to a textual entailment model from AllenNLP [9] and based on a technique called decomposable attention [10]. Features from that model are input to a multi-layer perceptron that provides the final classification. (See Figure 1.) While Lee et al. also investigate approaches that involve fine-tuning the language model, in this paradigm, only the multi-layer perceptron learns.

4.2 Masking

We evaluate five alternatives to Lee et al.'s approach of masking the last named entity (see Figure 2). Three involve masking tokens of different lexical categories, or different amounts of tokens: masking all entities, masking the last verb and masking all verbs. The other two are meant as baselines to evaluate how important the choice of token to mask is: masking a token picked at random, and masking a content token picked at random. We believe masking content tokens is more likely to test the LM's factual knowledge, with masking non-content being more likely to test linguistics knowledge.

To find named entities and detect tokens' parts of speech, we use a spaCy [11] model that considers dates to constitute named entities. That means that here, named-entity masking is analogous to salient span masking, a technique that has been shown to boost performance in pre-training LMs [12].

4.3 Token selection

We evaluate four alternatives to Lee et al.'s approach of taking the highest-scoring from the language model (see Figure 3). Three aim to extract more information, or more relevant information, from the LM. Those include selecting not just the top token but also the score provided by the LM, selecting the top three candidates instead of the top one candidate, and selecting the top candidate whose part of speech matches the token that was masked. (In the event that BERT's top five predictions did not include a token whose part of speech matched the input's, we defaulted to the highest-scoring token.)

These options are meant to address the problem of there being many "good" ways to fill in a missing word in a sentence. An LM might predict something linguistically plausible but distantly related to the masked token. By sampling more predictions from the language model, or constraining the

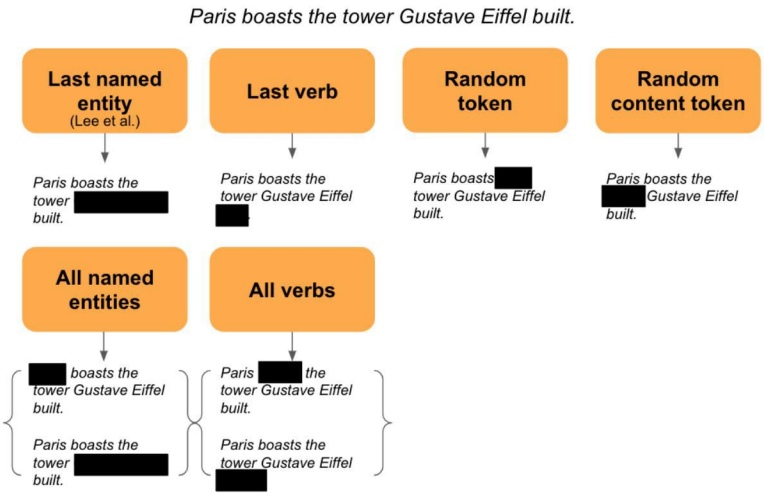

Figure 2: We evaluate six strategies for masking a claim before inputting it to an LM.

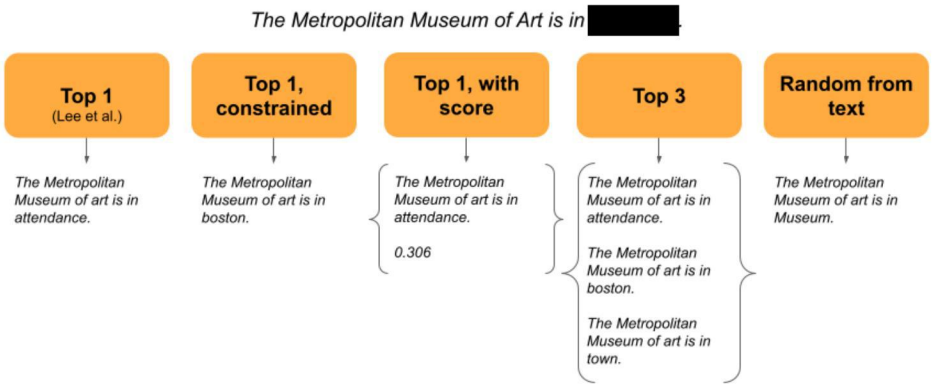

Figure 3: We evaluate five strategies for selecting tokens from an LM.

predictions to be more linguistically similar to the masked token, we hoped to extract more useful information for classification downstream.

We hoped that extracting the score from the language model, and not just the top candidate, would address one of the pipeline's weakest areas: its low performance on NEI examples. By including the probability assigned to predicted token, we thought we could provide information that would show the downstream classifier when the LM was uncertain about its prediction, which could happen because facts supporting or refuting the claim are not available.

Finally, as a random baseline meant to test the LM's importance, we also implement an alternative that ignores the LM altogether, and instead predicts a random token from the masked example.

4.4 LM removal

We also evaluate a version of the pipeline that removes the language model entirely, and the masking and token-selection processes (see Figure 1). The claim is sent directly, as both premise and hypothesis, to the entailment model. We call this pipeline NoLM.

5 Experiments

5.1 Data

We use the popular fact-checking dataset FEVER [2], which has almost 200,000 human-generated examples based on Wikipedia, with its standard split. The task we consider is to predict whether a natural-language claim is supported (SUPPORTS), refuted (REFUTES) or there's not enough information (NEI). Consider the following example from the training set:

1. **Claim:** "Nikolaj Coster-Waldau worked with the Fox Broadcasting Company."
2. **Label:** SUPPORTS
3. **Evidence:** [[[92206, 104971, "Nikolaj_Coster-Waldau", 7], [92206, 104971, "Fox_Broadcasting_Company", 0]]]

The evidence data, which is provided for SUPPORTS and REFUTES but not NEI examples, indicates which Wikipedia sentences justify the claim. We ignore this data, here, since we focus on extracting knowledge from an LM, not from a corpus. Some evaluations on FEVER also measure a system's ability to produce evidence, which we ignore as well, following Lee et al. While we recognize the value in providing justification, we leave this for future work.

5.2 Evaluation method

Canonical FEVER scoring is accuracy with the requirement that matching evidence be provided. We focus instead on vanilla classification accuracy, based just on the label; and also consider per-class precision, recall and F1-score.

5.3 Experimental details

We implemented our code in PyTorch [13]. We rely on off-the-shelf models for several components, but train our own neural net and implementing the masking and token-selection alternatives ourselves.

5.3.1 External models

The pipeline uses off-the-shelf models at several stages, following Lee et. al. The language model is BERT [8], provided by Huggingface [14]. (Lee et al. used BERT-Large, while we use BERT-Base for computational efficiency.)

The textual entailment model is from AllenNLP [9] and based on a technique called decomposable attention [10]. We modified the model's `forward` method to return its last hidden state, a 400-dimensional vector that serves as the input to the neural net.

To detect named entities and determine tokens' parts of speech, we use spaCy's [11] `en_core_web_sm` model. (We use the Universal Dependencies [15] list of open-class POS tags to determine what count as a content token.)

5.3.2 Neural net

We trained a multi-layer perceptron with one hidden layer, using a cross-entropy loss function. Like Lee et al., we used a 100-dimensional hidden layer, with a learning rate of 0.001, a batch size of 32, a max number of epochs of 200 and Adam as the optimizer.

5.3.3 Collating entailment features.

The all-entities and all-verbs masking strategies mask a different number of tokens in each example. That means we must collate a variable number of entailment features before passing them to the neural net. In the all-entities case, we tested four methods for doing so: 1) concatenating and padding, 2-3) applying an element-wise max or sum; and 4) using one neural net (optimized jointly with the net performing classification) to compress each 400-dimensional entailment vector into a 3-dimensional vector, and concatenating and padding these reduced-dimensionality outputs. That final approach was inspired by this post on training models to handle variable number of inputs [16]. We saw minimal

effect on performance between the techniques, and used the element-wise max method to evaluate our models.

In the setting where we select the top three tokens from the LM, we simply concatenate the resulting entailment features.

5.4 Results

We first provide the results of each masking strategy (see Table 1). They vary remarkably little. Not only does overall accuracy fluctuate by only about 1 absolute percentage point, but even the per-class precision, recall and F1 scores are highly similar. Our prediction that masking more tokens could extract more knowledge from the LM and lead to better fact-checking performance was not borne out.

We also provide the results of each token-selection strategy (see Table 2). Again, they vary remarkably little. Selecting the top-three candidates instead of one does not appear to extract additional knowledge that aids fact-checking classification. Constraining the part of speech of the predicted token does not appear to extract more relevant knowledge that aids classification. Selecting the score as well as the top token does not appear to improve performance on NEI examples. [1]

We iteratively added to the masking and token-selection strategies we explored to discover why this was the case. After seeing so little variation in performance between models that masked just one entity, and those that masked multiple entities in turn, we hypothesized that the system's overall performance had little sensitivity to the choice of token to mask. When we saw that picking a token to mask at random also had little impact, that supported this prediction. After seeing so little variation in performance between models that selected tokens from the LM differently, we hypothesized that the system's overall performance had little sensitity to the LM prediction. When we saw little change in performance from "mocking" the LM with a module that predicts a token chosen randomly from the input, that supported this hypothesis.

The near-invariant results led us to develop the NoLM pipeline, which ablates the language model (and with it, masking and token-selection) altogether. We were surprised to see that this pipeline too has comparable performance to Lee et al.'s system (see Table 3). While the per-class precision and recall values do fluctuate (most noticeably for NEI examples, where F1 drops from 0.34 to 0.27), overall accuracy varied little.

Table 1: Masking strategies

Model	Accuracy	Label	Precision	Recall	F1
Last named entity (Lee et al.)	0.48	REFUTES	0.63	0.45	0.52
		SUPPORTS	0.43	0.74	0.54
		NEI	0.48	0.26	0.34
All entities	0.49	REFUTES	0.65	0.45	0.53
		SUPPORTS	0.42	0.78	0.55
		NEI	0.5	0.23	0.31
Last verb	0.48	REFUTES	0.61	0.47	0.53
		SUPPORTS	0.42	0.72	0.54
		NEI	0.48	0.25	0.33
All verbs	0.48	REFUTES	0.64	0.44	0.52
		SUPPORTS	0.42	0.75	0.54
		NEI	0.47	0.24	0.32
Random token	0.48	REFUTES	0.64	0.43	0.51
		SUPPORTS	0.42	0.75	0.54
		NEI	0.47	0.26	0.33
Random content token	0.49	REFUTES	0.63	0.48	0.55
		SUPPORTS	0.43	0.71	0.54
		NEI	0.47	0.29	0.36

[1] Across this section, metrics are derived from evaluatuon on the FEVER dev set. While we sought to evaluate the NoLM pipeline on the test set, FEVER test-set evaluations are facilitated through Codalab and are rate-limited. We temporarily used up our allowance by submitting incorrectly encoded versions of our results. But given that the performance we see on the dev set closely matches what Lee et al. report, and the across-the-board consistency of the models' performance, we do not expect test-set evaluation to be significantly different.

Table 2: Selection strategies

Model	Accuracy	Label	Precision	Recall	F1
Top 1 (Lee et al.)	0.48	REFUTES	0.63	0.45	0.52
		SUPPORTS	0.43	0.74	0.54
		NEI	0.48	0.26	0.34
Top 1, with score	0.49	REFUTES	0.65	0.44	0.52
		SUPPORTS	0.42	0.76	0.54
		NEI	0.49	0.26	0.34
Top 3	0.48	REFUTES	0.57	0.5	0.53
		SUPPORTS	0.42	0.71	0.53
		NEI	0.48	0.21	0.3
Top 1, constrained	0.48	REFUTES	0.62	0.46	0.53
		SUPPORTS	0.42	0.76	0.54
		NEI	0.49	0.22	0.31
Random from text	0.48	REFUTES	0.62	0.44	0.52
		SUPPORTS	0.42	0.76	0.54
		NEI	0.49	0.24	0.32

The consistent results across different masking and token-selection strategies led us to perform an ablation experiment: Removing the language model, and the masking and token-selection processes (see Table 3). We were surprised to see that even this had little impact on performance. While F1 score on the NEI category declined, overall accuracy dropped only by about 0.9 absolute percentage points on removing the LM.

These figures remain well-below state-of-art on the FEVER dataset, but are comparable to a FEVER baseline which has steps for selecting documents, selecting sentences and textual entailment [2] (see Table 3). Having thought of the LM as the portion of the pipeline responsible for factual knowledge, we found it remarkable to see how little impact on performance came from different ways of using the LM, including forsaking it altogether.

Table 3: Pipeline, with and without a language model

Model	Accuracy	Label	Precision	Recall	F1
Lee et al. (our implementation)	0.48	REFUTES	0.63	0.45	0.53
		SUPPORTS	0.43	0.74	0.54
		NEI	0.48	0.26	0.34
No LM	0.48	REFUTES	0.67	0.43	0.52
		SUPPORTS	0.41	0.81	0.54
		NEI	0.5	0.18	0.27
2018 competition high score [17]	0.68				
Current Codalab leaderboard high score [18]	0.79				
FEVER baseline [17]	0.49				

6 Analysis

In this section we seek to answer two questions. Why does the use of a language model in this paradigm seem to have little impact on fact-checking performance? How can NoLM, with no corpus or knowledge base, but just an entailment model and neural net, attain the perforamance it does?

6.1 Language models and fact-checking

Even though BERT was pretrained on text including more than 2 billion words from Wikipedia [8] — the same source which humans used to generate FEVER claims — removing BERT, or trying to "use it more" through querying it more extensively, have little impact on accuracy. This is surprising given our expectation that the language model was responsible for providing the world knowledge in the setup.

We offer several reasons it may be difficult for LMs to contribute in this set up without finetuning.

The foremost challenge may be that there are often many plausible ways to fill in a blank in a sentence. That can mean that BERT's prediction for a masked token can conflict with a true claim. But BERT often predicts high-probability tokens — for example, predicting pronouns after a named entity is mask —which may be underinformative. From the FEVER test set, consider "Grace Jones is a

dancer," masked as "[MASK] is a dancer," for which BERT predicts "she." That gives an entailment model little to go off of, especially compared to fact-checking systems that have access to corpora or knowledge bases. And constraining the part-of-speech of the predicted token to match the masked token, as we explored in one strategy, may actually worsen this dynamic. For the same example, the top-scoring named entity BERT predict is "kim," semantically even further from "Grace Jones."

We considered only the zero-shot setting here for BERT, refraining from finetuning so as to test the knowledge already within the model. Fine-tuning BERT can lead to stronger fact-checking performance, as Lee et al. show and [14] suggests.

Additionally, language models are also known to be sensitive to the phrasing of a query [19], and while we considered different methods of masking, we didn't consider rewording queries. Consider the example from Figure 3: BERT's top three predictions for "The Metropolitan Museum of Art is in [MASK]." are "attendance," "boston" and "town," which are false or nonsensical. Its top prediction for "The Metropolitan Museum of Art is located in [MASK]." is "manhattan," which is correct. In these experiments, we explored different options for masking portions of the claim, but not different ways of phrasing the claim as a query. While we don't see obvious reason to suspect that the wording of FEVER claims specifically would challenge BERT in particular, further research could evaluate whether claims to fact-check can elicit better knowledge from BERT.

6.2 NoLM's performance

We invite further research, too, on the strength of the pipeline consisting of just the decomposable attention textual entailment model and the neural net. Entailment models are often components of fact-checking systems — and there are reported benefits of decomposable attention in particular [20] — but they almost always draw input from documents or passages. In NoLM, the source of factual knowledge is unclear.

With an identical premise and hypothesis, the entailment model predicts entailment, as expected, exceedingly often. (We found rare examples for which the model, through the Python interface but not the web one, predicted contradiction. From the FEVER dev set, try "The Fame was released in February 2016.") But in taking the last hidden state from the model, we essentially use the model to create rich 400-dimensional representations of the input. Vector representations of language can be said to capture world knowledge, like how word vectors can complete analogies [21]. Even with it being designed for natural language inference, the textual entailment model may capture world knowledge that manifests in this last hidden state.

Further ablation experiments may seek to answer to what extent the neural net, even with its single hidden layer, is responsible for the performance, and to what extent the credit goes to the textual entailment model.

7 Conclusion

Inspired by recent work seeking to leverage LMs and their factual knowledge for fact-checking, we sought to explore how different ways of using the LM could boost fact-checking performance. Evaluating models based on the FEVER dataset, we saw little change in performance from different methods of masking an input to claim to query an LM, and from different methods of selecting tokens from the LM. In fact, even removing the LM from the pipeline altogether had little impact on performance.

We sought to illustrate challenges in using LMs for fact-checking in this paradigm, and investigate how the language-model-less version of this pipeline maintain its performance. We develop hypotheses about how the system, now based just on an off-the-shelf textual entailment model and a neural net, may attain better-than-random fact-checking performance. While the performance of this technique remains below state-of-art, we hope to contribute to improved fact-checking systems, and to add to understandings about extracting knowledge from LMs and textual entailment models.

Acknowledgments

We are grateful to Nayeon Lee for their help in understanding the implementation of their system, and to our CS 224N mentor Megan Leszczynski for ideas about debugging an initial implementation of the model, and understanding the performance of the NoLM setup.

References

[1] Nayeon Lee, Belinda Z. Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online, July 2020. Association for Computational Linguistics.

[2] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 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, June 2018. Association for Computational Linguistics.

[3] Vamsi Krishna Pendyala, Simran Sinha, Satya Prakash, Shriya Reddy, and Anupam Jamatia. Validation of facts against textual sources. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 895–903, Varna, Bulgaria, September 2019. INCOMA Ltd.

[4] Andreas Vlachos and Sebastian Riedel. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA, June 2014. Association for Computational Linguistics.

[5] James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[6] Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? *CoRR*, abs/1909.01066, 2019.

[7] Mayank Jobanputra. Unsupervised question answering for fact-checking. *CoRR*, abs/1910.07154, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018.

[10] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *EMNLP*, 2016.

[11] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[15] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal dependencies v2: An evergrowing multilingual treebank collection, 2020.

[16] Andre Holzner. Learning a function with a variable number of inputs with pytorch, Feb 2018.

[17] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. The fact extraction and verification (FEVER) shared task. *CoRR*, abs/1811.10971, 2018.

[18] Competition.

[19] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *CoRR*, abs/1911.12543, 2019.

[20] Nayeon Lee, Chien-Sheng Wu, and Pascale Fung. Improving large-scale fact-checking using decomposable attention models and lexical tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1133–1138, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[21] Carl Allen and Timothy M. Hospedales. Analogies explained: Towards understanding word embeddings. *CoRR*, abs/1901.09813, 2019.