Fact Checking Task: Do LLM Sentence Embeddings Encode Real World Knowledge?

Ben Wilen and Kelsey Richter
Williams College - CS375
Natural Language Processing

Abstract

Recent research around the development of large language models (LLMs), including the development of OpenAI's ChatGPT and Google's Bard, have illustrated the ability of LLMs to store real-world knowledge in their parameters. Other recent work has specifically focused on whether the real-world knowledge of LLMs can be used to fact-check statements. While previous work has focused on how LLMs compare to other machine learning models for the fact-checking task, we focused on whether the number of parameters in a LLM is correlated to the ability of a model to encode real-world knowledge, and whether the fact-checking task should be treated as an ordinal regression task or regular classification task. We show that the number of parameters do not necessarily correlate to the ability for a model to encode real world knowledge, but instead the architecture and training corpus of the model might have more of an impact. We also show that treating this classification as an ordinal regression task did not increase the accuracy of the models. Although our models do not show conclusive results about which model architectures best encode real-world knowledge, our experiments illustrate that LLMs are a viable option for fact-checking, and that more exploration is required to determine how to maximize real-world knowledge inside LLM parameters.

1 Introduction

In recent years, the fact checking task has gained popularity as news outlets have attempted to fact check public figures, governments, and other news sources. A major area of natural language processing (NLP) is question-answering, and fact checking is an extension of question answering. Progress surrounding the use of pretrained language models has led to various improvements for downstream NLP tasks (Petroni et al., 2019). Some of these advancements include the tasks of question-answering, commonsense reasoning, and semantic relatedness. In addition to linguistic knowledge, these models may be learning relational knowledge, and thus, have the potential to serve as knowledge bases. Language models as representations of relational knowledge presents the possibility for the use of language models to answer queries. This has many advantages over structured knowledge bases. Language models eliminate the need for information retrieval. Additionally, they require no human supervision (Petroni et al., 2019). The goal of our research is to examine how well sentence embeddings generated by large language models encode real world knowledge, and specifically the validity of the statement they represent. To do so, we looked into the use case of fact checking statements.

Fact-checking is the task of assessing the truthfulness of claims made by public figures, which is commonly performed by journalists. The formal task definition of fact checking is "the assignment of a truth value to a claim made in a particular context"(Vlachos and Riedel, 2014). Thus, fact-checking can be viewed as a classification task. But, statements in real-world contexts are often not entirely true or false. Therefore, we use a data set that contains six labels—"true", "mostly-true", "half-true", "mostly-false", "false", and "pants-fire". We hypothesized that it is best to consider fact-checking as an ordinal classification task. The time consuming process of Fact-checking is currently done manually by journalists. The potential to harness fact-checking capabilities using NLP would allow for the automation of these processes. Additionally, ordinary citizens could benefit from the increased ability to fact-check the information that they come across daily, such as on platforms like Twitter (Vlachos and Riedel, 2014).

We performed a fact checking task using three different language models in order to examine their effectiveness. In performing this comparison, we sought to examine whether the number of parameters in a LLM is correlated to the ability of a model to encode real-world knowledge. We used three models, t5-small, BERT, and t5-large, which are listed in the order of increasing parameters. Additionally, we used three methods of classification to perform the fact-checking task in order to gain insight into whether fact-checking should be treated as an ordinal regression task or regular classification task. The three classifier methods we implemented were logistic regression, ordinal regression, and a deep neural network.

2 Related Work

Thorne et al., 2018 sought to find the state-of-the-art fact checking model. Competitors submitting models used mainly read-and-retrieve models, which contained information retrieval from open-domain systems and reading comprehension of the top documents. Models then typically used entailment to determine whether selected evidence supported the claim. However, in 2019, Petroni et al.'s (2019) work sought to determine how much relational knowledge is present in "pretrained off the shelf language models" such as ELMo and BERT. They posed the question of how much knowledge is stored and, further, how this knowledge differs among different knowledge relations, such as facts about entities, common sense, and general question answering. The authors explain that the existing research surrounding language models focuses on understanding linguistic and semantic properties of word representations. They instead sought to compare the ability of pretrained language models to store factual and commonsense knowledge with standard relation extraction approaches. They use what they call the LAMA (LAnguage model analysis) probe to test the factual and commonsense knowledge capabilities of language models. It consists of a set of knowledge sources, each of which is comprised with a set of facts. The facts are either subject-relation-object triples or question-answer pairs. The facts are converted into cloze statements, which are then used to query the language model for the missing token. The authors define the success of a pretrained model by the task of masking: whether it can successfully predict the missing word from a sentence. The various types of knowledge are tested by using various knowledge sources, including relations between entities stored in Wikidata, common sense relations between concepts from ConceptNet, and knowledge needed to answer natural language questions in SQuAD. The results yielded that the largest BERT model does have a comparable ability to capture relational knowledge to that of non-neural and supervised alternatives. They found that BERT-large consistently outperforms other language models in recovering factual knowledge, common sense knowledge, and is more robust when it comes to the phrasing of a query. The authors predict that this finding can be attributed to the large amount of data that BERT-large has processed compared to other models. This connects to our goal of determining whether the number of parameters for each language model affects their performance. Their findings suggest that, while relation extraction performance might be difficult to improve with more data, language models trained on huge corpora might become a viable alternative to traditional knowledge bases extracted from text in the future.

Roberts et al., 2020 further found that the model size positively correlated with the ability for a model to store real world knowledge. However, they noted that they only began to see state-of-the-art results at 11 billion parameters, more than 10 times larger than the models we used in our experiments.

Vlachos and Riedel, 2014 introduce the task of fact-checking and detail the construction of a dataset that contains statements fact-checked by journalists. They define fact-checking as an ordinal classification task. The dataset was constructed using statements from two popular fact checking websites, the fact checking blog of Channel 45 and the Truth-OMeter from PolitiFact. The labels of the verdicts were aligned to a five-point scale: TRUE, MOSTLYTRUE, HALFTRUE, MOSTLYFALSE and FALSE. The authors highlight the difficulty of the task. Fact-checking is related to the tasks of textual entailment and machine comprehension, but differs because the text that should be used to predict the correct answer is not given as input. Instead, the sources for that information must be located to predict the verdict label.

Iyyer et al., 2014 sought to apply deep learning to question answering, rather than the previous manually defined string matching rules and bag of words representations. The previously mentioned methods are ineffective when question text contains very few individual words that are indicative of the answer. The proposed recursive neural network (rnn) model can reason over such input by modeling textual compositionality.

The main study that inspired our own research was the first study to examine whether the knowledge of LMs could be leveraged for fact-checking. Lee et al., 2020 expanded on the above research of LLMs using real world knowledge for question-answering, and specifically proposed that it could be used for fact checking. Their model consisted of entity masking, entity prediction using BERT or another LLM, entailment for verification, and finally a multi-layer perception network to predict the final class. This paper achieved 38 percent accuracy, which was better than the random baseline but worse than the state-of-the-art FEVER read-and-retrieve models. However, they used the FEVER data set which only included true, false, and not enough information classes.

3 Dataset

We used the Politifact Fact Check Dataset. The dataset contains 21,152 statements that are fact checked by experts. All of the statements are collected from the popular fact check website PolitiFact. Each statement has a categorization into one of 6 categories: true, mostly true, half true, mostly false, false, and pants on fire. Each record consists of 8 attributes: the verdict, the statement originator, the statement, the statement date, the source of the statement, the individual who fact checked the statement, the date of the fact checked article, and a link to the article. These attributes can be seen in more detail in table 2. For the purposes of our work, we only used the verdict, statement, statement originator, and date attributes. We divided the data into training, dev, and testing sets, 1,000, 200, and 200 respectively per class. In order to clean the dataset, we removed all statements from the set from the year 2022, given that the models we used are not up to date enough to accommodate for that information.

4 Methods

4.1 Pipeline

As seen in Figure 1, we use a 3-step pipeline. Given a statement to be fact-checked, the first step of our pipeline is to generate a statement dense embedding. To do so, we pass the embedding through a LLM, and generate a sentence dense embedding by retrieving the [cls] token embedding for BERT

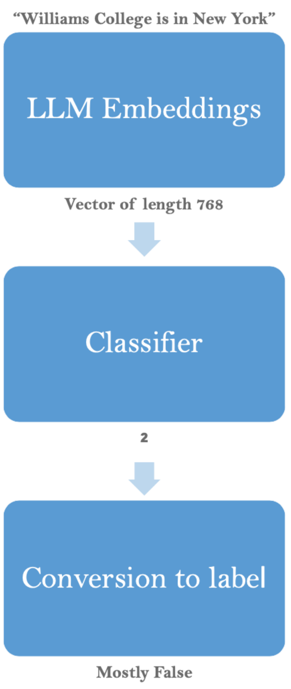

Figure 1: pipeline

and by retrieving the first decoder output embedding for T5. In step two, we pass the embedding into a classifier which generates a label 0-5 corresponding to each of the six Politifact categories. In step three, we pass the integer into a simple switch statement function which converts the label into its word representation (e.g. "true").

4.2 Experiment Design

We used 3 LLMs for statement dense embeddings (t5-small, BERT, and t5-large) and 3 classifiers (logistic regression, ordinal regression, and a deep network) in our experiment. Doing so would allow us to determine which steps in our pipeline were responsible for fluctuating the accuracy of our model (e.g. if all three classifiers return similar accuracies, we could determine that the LLM was more correlated to the accuracy of the model). Using this synchronous approach also allowed us to simultaneously study which LLMs best encode real-world knowledge, and which classifiers handle the fact checking task the best.

We chose the 3 LLMs given their accessibility, usability, and because they each had a different number of parameters. The BERT model we used has 109,482,240 parameters. Goldberg, 2019 did work to access BERT's syntactic abilities and found that BERT models perform very well on all of the syntactic test cases. We also believed that the bidi-

Classification	Number of Statements
false	5625
half-true	3597
mostly-false	3432
mostly-true	3332
pants-fire	2703
true	2463

Table 1: The number of statements that correspond to each classification in the dataset.

Attribute	Attribute Description
verdict	The verdict of fact check in one of 6 categories {true, mostly-true, half-true, mostly-false, false, and pants-fire}
statement_originator	the person who made the statement being fact checked
statement	statement being fact checked
statement_date	the date when statement being fact checked was made
statement_source	the source where the statement was made.
factchecker	name of the person who fact checked the claim
factcheck_date	date when the fact checked article was published
factcheck_analysis_link	link to the fact checked analysis article

Table 2: Descriptions of the attributes in the data set

rectional encoding of BERT would help ensure that all tokens were included in the embeddings (e.g. one word in a statement can drastically change its validity).

Ni et al., 2021 found that T5 competitively compared to other encoder-decoder models. They used an encoder model with a contrastive loss function, which moves embeddings closer and farther from each other given the sentence similarity. This paper concluded that the T5 models with more parameters performed better. We used a T5-small model with 60,506,624 parameters and a T5-large model with 737, 668, 096 parameters.

We then chose 3 classifiers. We first choose to use linear regression because it is one of the most common (and simplest) classification models, and can be used as a psuedo-baseline for classification models. We used Scikit-Learn's linear regression model. The second classification model used was a neural network. Given the complexity within sentence embeddings, a neural network would be able to detect more complex patterns and feature relations than linear regression would. We built a neural network using PyTorch. The model consists of 2 hidden layers and uses a cross-entropy loss, which is standard for a deep learning classifier. Ordinal regression allows for classification when predicting an ordinal variable, which is a variable whose value exists on an arbitrary scale, where the relative ordering between values is significant. We predicted that ordinal regression would result in performance enhancements for our fact-checking task because of the nature of the fact-checking classification rank. The classification is scored on a scale from 0 to 5, so the relative ordering between them should be crucial to determining accuracy. For example, if a statement's correct label is true, the result mostly-true should be viewed as more accurate than the result false, even though both are incorrect. We used statsmodels ordinal model.

We run our code below nine times, once for each combination of LLM and classifier. Future exploration should include repeating this experiment multiple times to validate the results.

4.3 Hyper-parameter tuning

For all three models, we performed cross validation to determine which hyper-parameters to use. We used a 5-fold cross validation, averaging across the folds. For the deep network, we cross-validated for learning rate, dropout probability, and hidden layer dimensions. For linear regression, we cross-validated for regularization weight. Finally, for ordinal regression, we cross-validated for whether the model should use a logistic or probit underlying distribution.

4.4 Code

The codebase is separated into four files, a FactCheck python file, Statement Embeddings python file, Statement Classifier python file, and Deep Network Python file. All code can be run through the FactCheck file which serves as the interface into the pipeline. Prior to initializing a FactChecker class, we stored statement embeddings for the Politifact Fact Check dataset for each of the LLMs. To do so, we ran the storeEmbeddings functionality within the Statement Embeddings file for each of the LLMs (t5-small, BERT, and t5-large), which generated 3 JSON files per model (training data, dev data, and testing data). To store the embeddings, each statement is first modified to include the statement's speaker, and then is passed through the LLM to generate an embeddings. Because the LLMs we use are sequence-to-sequence models, we retrieved the [cls] token's embeddings for BERT or the first decoder output embedding for T5.

When a FactChecker class is initialized, the JSON files storing the data corresponding to the correct LLM (given as a parameter to FactChecker) are retrieved from disk. However, by calling trainEmbeddings(), the user can regenerate the json files. A user will then call trainClassifier(), which will train the given classifier model on the statement dense embeddings and corresponding labels. To do so, the Statement Classifier class initializes that specific model, and then trains/fits it to the training data. Finally, the user can call get_training_data_accuracy(), which returns the accuracy of the model on training data. We used that function to generate our results. We repeated this process for each of the nine combinations of LLMs and classifiers.

Lastly, the FactCheck function serves as a users entry point into a fully trained model. They can input any statement, and it will generate a dense embedding of the statement, predict its label through the pretrained classifier, and convert that label to its word representation.

4.5 Evaluation Metrics

To evaluate the performance of the classifiers, we simply recorded their accuracy. This was calculated by taking the average of the number of times that the predicted classification by the model matched the given classification of a statement in the data set.

5 Results

Our results showed similar performance across all three language models. The accuracies across the different models for each classifier all range from about 0.24 to 0.285 (figure 2). Given that there are a total of six classification categories, results that reflect the models showing no knowledge and predicting based on chance would have accuracies of about 0.17. Therefore, we can conclude that our models did encode some real world knowledge given that they each outperformed what we would expect to see if their predictions were based on chance.

The accuracies of each Language model, for each type of classifier, are shown in figure 2. From this data, we can see that BERT outperformed both t5 models. Across all three classifiers, the highest accuracy was observed when using BERT as the pretrained language model. Additionally, we can see that the deep network performed the most poorly across all three language models. However, this result could potentially be explained by the limited amount of training data used to train the model. Neural networks typically require a substantial amount of training data to not over-fit to specific examples. Ordinal regression and logistic regression performed similarly across the various language models. Logistic regression did the best on t5-small and BERT, whereas ordinal regression did the best on t5-large.

Given that all three classifiers performed similarly (within 5 percent), we believe that the inaccuracy of the models is more attributed to the LLMs. We hypothesize that the models are not large enough to encode real-world knowledge specific to the statements the models were trained on. Also they may have been outdated for more recent statements. In a closer examination of our data, we saw that many of the statements that were in the Politifact data set included niche statements, such as statements said by lesser-known individuals and statements about very specific details. We believe that the models we used, given their relatively small number of parameters, are unable to store real-world knowledge that is that specific. Furthermore, the models we used might have been trained on a corpus that did not include knowledge of these niche statements. Finally, an examination of our fact checker pipeline illustrated that, for example, a statement such as "Barack Obama was the President of the United States" returned false, but "Barack Obama was the 45th President of the United States" returned true. We believe that further examination is required to determine how small difference such as the one above appear in statement embeddings.

In doing this experiment, we sought to examine whether the size of a language model would determine its effectiveness to encode real world information. The results that BERT had the highest performance among the language models suggest that this is not the case. The exact number of parameters can be seen in figure 2, but BERT has significantly less parameters than t5-large, so the number of parameters did not translate to better performance. Further, we can see from the graph that t5-small outperformed t5-large. One potential explanation for this result that could be a direction for future research is that the architecture of the language model matters more for performance and ability to encode real world knowledge, given that the BERT model outperformed both t5 models, regardless of parameter size. Another potential explanation is that the BERT model was trained on data more relevant to the Politifact dataset.

Also to note, we hypothesized that ordinal regression would be the most effective classifier for the task of fact checking. However, this did not end up being the case. Ordinal regression did not show significant improvements in performance over standard logistic regression. In figure 3, the accuracies across each classification for the three classifiers are shown. We can see from this data that all three classifiers had significantly higher performance for the edge cases of fully true or fully false, "pants fire" statements. And in contrast, the middle classifications such as "half true" and "mostly true" have lower accuracies. Intuitively, this makes sense, given that the middle classifications are more arbitrary and are more likely to have overlap. We expected ordinal regression to improve the accuracy of these middle classifications; however it performed similarly to the other models, showing no significant improvement and even doing worse in some cases.

As expected, all three classifiers performed better for "true" and "pants-fire", suggesting that it is easier to classify the boundary classes than the intermediary ones. We specifically thought ordinal

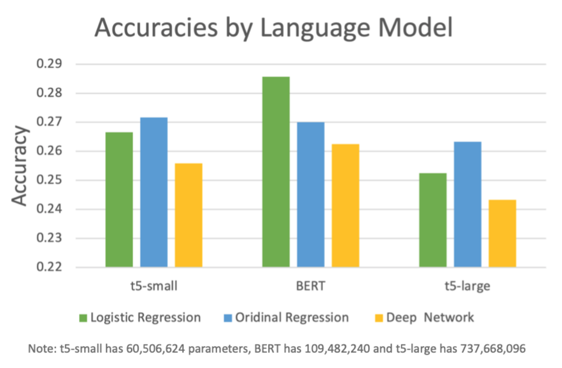

Figure 2: Accuracies by language model across the classifiers

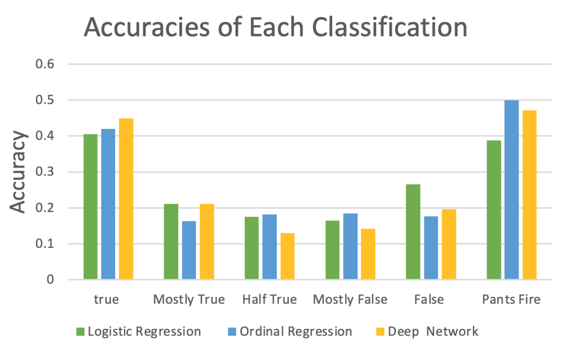

Figure 3: Accuracies of each classification type across the classifiers

regression would be more accurate for the middle classes, and as figure 3 illustrates, that was not the case. Further research should explore that distribution of labels for each label to determine whether the classifiers resulted in a disproportionate number of one label versus another.

Conclusion

We did not find any significant results suggesting that the size of a language model affects its ability to encode real world information. However, we believe that BERT's architecture impacted its ability to more accurately encode real-world information, suggesting that a models architecture might be tied into ability to encode real-world knowledge in its parameters. However, an interesting area for future work would be repeating our procedure, but using significantly larger language models to encode the statement embeddings. As mentioned in the related work section, research suggests that we may not be able to capitalize on the ability of large language models to store real-world knowledge until they are as large as 11 billion parameters.

Given that our results indicated ordinal regression had no impact on classification, we think that ordinal classification should be replicated with more data and larger LLMs for statement embeddings. Further research could also study using an ordinal regression loss function in a deep learning network, utilizing both the ability of a neural network to detect complex patterns and ordinal regression's impact of correctly assessing ordered labels. Our work is one of the first to use ordinal regression for the fact checking task, and although our results do not show a benefit to it, we still believe that ordinal regression can be beneficial in transitioning the fact checking task from binary labels (true/false) to multi-class labels.

Despite our model's accuracy of .25-.28, we believe this is a promising result. This significantly surpasses the random baseline of .17 by approximately 10 percent while Lee et al.'s (2020) model surpassed the random baseline by 5 percent. Our research is one of the first studies to use a six label fact checking data set, illustrating the difficulty of large language models to make nuanced class distinctions. We believe our results can serve as a baseline for future multi-class fact checking models.

Although our results illustrate that more advanced and larger models might be capable of accurately performing the fact checking task, we acknowledge that LLMs aren't perfect sources of knowledge, and can hallucinate results. If LLM fact checkers are used in industry, they should be used with caution, and advise users of the underlying systems making the classifications.

Nevertheless, the power of LLMs to encode real-world knowledge in embeddings is an impactful result, and has the potential to both supplement fact checking, question answering, and many other knowledge based tasks.

References

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar. Association for Computational Linguistics.

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

Rishabh Misra. 2022. Politifact fact check dataset.

Rishabh Misra and Jigyasa Grover. 2022. Do not 'fake it till you make it'! synopsis of trending fake news detection methodologies using deep learning. In *Deep Learning for Social Media Data Analytics*, pages 213–235. Springer.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models.

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

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Andreas Vlachos and Sebastian Riedel. 2014. 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. Association for Computational Linguistics.