6.864 Final Project - Investigating Language Models as Knowledge Bases

William Chen Shreya Gupta Joseph Morales
Dept of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
{verityw, shreyag, jrales}@mit.edu

Abstract

Prior work has left questions unanswered regarding the ability of language models to extract semantic information and act as knowledge bases. It is currently unclear the extent to which models learn abstract representations of semantic relations rather than learning statistical correlations. We present an examination of the pre-trained Bidirectional Encoder Representations from Transformers (BERT) language model as a knowledge base. In this paper, we test the limits of pretrained BERT using the LAnguage Model Analysis (LAMA) probe and subsequently utilize that information for further exploration. This exploration includes attempting to inject new knowledge into the BERT model by training it on an updated LAMA database and testing the level to which BERT overfits by training it on a randomized LAMA database. Ultimately, we find that BERT's cloze question-answering abilities largely reflect the distribution of question types used to train/fine-tune the model – underrepresented question styles were usually answered more poorly. We thus conclude that although language models trained through conventional means are capable of learning new information, they are currently ineffective as knowledge bases because they do not effectively retain information that is not represented in the most recent distributions used to train them.

1 Introduction

Language models trained on large corpuses have been shown to extract linguistic information and structure from training data, which has proved beneficial for many practical NLP tasks. These include speech recognition, question answering, and machine translation (Kuo et al., 2002; Rajpurkar et al., 2016; Lembersky et al., 2011). However, language models' ability to extract semantic information, thus acting as knowledge bases, is an active area of research (Petroni et al., 2019). Specifically, it is unclear if these models simply learn statistical correlations between terms or if they develop more abstract representations of semantic relations. Moreover, knowledge from training data is effectively encoded into a language model's weights and biases. However, as these quantities generally do not change after training, they are not regularly updated as new information appears. The ability to tractably update the information stored within a language model neural network is still being researched (Wang et al., 2020).

In this paper, we explore the usage of the pre-trained Bidirectional Encoder Representations from Transformers (BERT) language model as a knowledge base. Specifically, we tested the limits of the cloze-answering capabilities of pretrained BERT by probing it using the LAnguage Model Analysis (LAMA) probe. We subsequently fine-tuned BERT on a significant subset of the entire LAMA dataset and evaluated the performance of both the base BERT model and the fine-tuned BERT model. We found that compared to the base BERT model, the fine-tuned BERT model's performance reflected the dataset populations, performing better on questions from sub-datasets that were overrepresented. After developing a fine-tuned BERT model, we proceed to two separate experiments. The first experiment is examining if we can inject new knowledge into BERT. We created an "updated" LAMA data set where we systematically replaced the answers of certain cloze questions and then retrained both the base BERT model and the fine-tuned BERT model on. An evaluation of these models demonstrated that the pretrained model, which had not seen the questions used in the updated dataset prior to updating, performed better than when we start with the fine-tuned model. The second experiment examined the extent to which BERT learns knowledge over memorization by randomizing the cloze statement-answer pairings in

a LAMA set and then retraining our base BERT model on it. Subsequently, we evaluated its performance and saw that the language model largely failed to memorize the nonsensical dataset's answers.

2 Related Works

One of the earlier attempts at quantitatively evaluating language models as knowledge bases is Petroni et al. (2019). Specifically, they constructed the LAMA (LAnguage Model Analysis) probe, which drew questions from four different existing natural language question-answer datasets (SQuAD, Google-RE, T-Rex, and ConceptNet (Rajpurkar et al., 2016; Orr, 2015; Elsahar et al., 2018; Speer et al., 2018)) and modified them into a unified form that transformers can try to answer. Specifically, as BERT found extensive success with the Masked Language Model (MLM) self-supervised training objective, wherein it is tasked with predicting masked-out words in a given text, it naturally lends itself to cloze question answering, wherein facts have a vital part blanked out and the answerer is tasked with filling it in (Devlin et al., 2019). Thus, the developers of LAMA chose to convert questions from the aforementioned datasets into cloze questions, e.g. "The theory of relativity was developed by _." With LAMA, Petroni et al. (2019) found that, even without fine-tuning, pre-trained transformers were often able to successfully answer these questions with relative success.

One disadvantage of this approach is that the language models are exclusively working with token-level answers. This naturally limits both the questions and answers that models can encounter and give, also narrowing the scope to single-token entities and concepts. Thus, Xiong et al. (2019) presented a new dataset and self-supervised training scheme called Weakly-supervised Knowledge-pretrained Language Models (WKLM) which forces entity-level predictions and reasoning by extracting text from Wikipedia as ground-truth positive facts and using Wikipedia anchor links to determine names and aliases of entities (which can be multi-token). Similar to the BERT and LAMA MLM training objectives, models are again tasked with predicting masked terms, though this time, entire entities are masked out.

Despite these successes, there has been controversy as to whether datasets like LAMA are valid probes of if language models are encoding abstract knowledge and reasoning representations or if they are merely using statistical correlation. Cao et al. (2021) investigated the three main ways in which MLMs are queried via cloze questions for knowledge retrieval, discussing the shortcomings of each:

1) *Prompt-based Retrieval - e.g."Steve Jobs was born in _"*: Cao et al. (2021) found that a given network's predicted answer distribution was undesirably similar between different datasets, even when the ground truth answers' distributions were vastly different. Further, the prompt phrasing itself heavily biased the predicted answers. All this suggests that the network is overfitting to certain prompt structures and dataset answers.

2) *Case-based Analogy - e.g. "Obama was born in Hawaii. Jobs was born in _"*: These analogy questions helped models predict the category of the answer, but did not consistently improve overall performance rates. In the above example, the model might be more likely to predict states, yet still not likely to correctly answer "California."

3) *Context-based Inference - e.g. "Jobs lives in California. Jobs was born in _"*: Often the context implicitly or explicitly gives away the answer, making prediction significantly more trivial, or else is completely unhelpful.

While Cao et al. (2021) provided significant quantitative evidence of language models' failures as knowledge bases, Razniewski et al. (2021) explored overarching qualitative concerns. While language models had certain advantages, like not having a fixed knowledge schema and often encoding unconventional facts/relations that knowledge base engineers may not think to include, they also have a host of disadvantages, like lack of sources, limited disambiguation, using statistical correlation (vs. referencing explicit knowledge), and as the present paper touches upon, inability for easy expansion/update of facts.

Nevertheless, Razniewski et al. (2021) acknowledged that hybrid approaches integrating neural language models with explicitly-engineered knowledge base-like structures may be successful. For instance, Logan IV et al. (2019) used knowledge graphs as a structured representation of information for neural networks to attend to, Wang et al. (2020) used external memory bank systems for information updating, and Petroni et al. (2020) (same lead author as Petroni et al. (2019)) used various information retrieval techniques to provide the language model with context from outside sources.

Despite these shortcomings, we wish to explore BERT's knowledge base capabilities in further detail. We specifically consider fine-tuned BERT's performance with the LAMA dataset (which the original authors did not consider) and look at methods for information updating and understanding.

3 Methods

3.1 Setup

3.1.1 Initial Installation

Our experiments center on the LAMA probe, so we started by installing and testing said software on both a Google Colab IPython notebook and a local machine concurrently. For local usage, we created a virtual environment containing all the Python library and module dependencies listed in the `requirements.txt` file in LAMA's repository. Some libraries were out of date or in conflict with other requirements, so we manually resolved them and generated a new `requirements.txt` file.

3.1.2 Hardware and Software Specifications

For our local tests, we used a Razer Blade 15 laptop running Ubuntu 18.04 with an NVIDIA GeForce RTX 3070 GPU installed for neural network training and inference. This laptop also had CUDA version 11.2 installed for PyTorch's GPU compatibility.

For our cloud-side implementation, we used Colab Pro to expand the available RAM pool, as several of our experiments were memory-intensive, crashing when Pro was not used.

3.1.3 LAMA Modernization

As LAMA has remained largely unchanged since Petroni et al. (2019), it used the old `pytorch-pretrained-bert` library models, which, as the name suggests, are PyTorch-adapted versions of the pretrained BERT models present in the HuggingFace `transformers` library (Wolf et al., 2020). As there have been extensive updates to the latter library, we decided to overhaul LAMA to use the `transformers` implementation, thus installing both the library and its dependencies. The port likewise required us to change some tokenization code specific to `pytorch-pretrained-bert` to be in-line with `transformers`' specs.

3.1.4 Models and Datasets

Due to our limited computational resources, we chose to use the base-size Bidirectional Encoder Representation from Transformers (BERT) model (cased) (Devlin et al., 2019), downloading the pretrained implementations from the HuggingFace transformers library. In terms of datasets, we downloaded the LAMA question-answer corpora, which were divided into the four original datasets which they were adapted from: SQuAD, Google-RE, T-Rex, and ConceptNet (Rajpurkar et al., 2016; Orr, 2015; Elsahar et al., 2018; Speer et al., 2018). The distribution of questions is shown in Table 1. Note the large difference in question population and length between the various datasets. Moreover, also note that these numbers seem to differ from the subset fact numbers displayed in Petroni et al. (2019) – while some of the minor differences can be accounted for due to how we counted the questions (we created a dataloader, described below, with batch-size of 1 to iterate through the entire dataset, so we only looked at questions that could be loaded by said construct), many of the other differences are unaccounted for. For example, ConceptNet was counted as having over 20,000 questions (with the data file containing over 29,000, though many are invalid for our dataloader), yet Petroni et al. (2019) states that it only contains 11,458.

3.2 Custom Dataset and Dataloader

To load questions for training and validation, we designed a custom dataset and dataloader. As Petroni et al. (2019) did not design a unified format for each of the four datasets comprising LAMA, we had to handle loading each one separately, as follows:

SQuAD and ConceptNet These two datasets are formatted the most simply. Upon being loaded as dictionaries from `.jsonl` file, a given question has keys `masked_sentences` and `obj_label`, containing a length-one list with the cloze question (with the blanked out word replaced with "[MASK]") and a string representation of the masked word respectively. These are both simply yielded.

Google-RE This dataset's questions are saved in a similar format to SQuAD and ConceptNet, except the `masked_sentences` list contains more than one sentence (question), so the dataset object yields each one contained

	LAMA Subset				
	SQuAD	**TREx**	**ConceptNet**	**GoogleRE**	**Total**
Number of Questions	305	1254595	23324	5572	1283796
Average Words per Question	9.97	29.37	9.63	20.38	28.97

Table 1: Various metrics of the LAMA subsets.

in its own list separately (all of which share the same answer of `obj_label`).

TREx Finally, TREx is saved very differently from the other three datasets. Each line of the `.jsonl` contains many questions, all with the same theme (as opposed to each line being one question or a few questions with the same answer). These questions are contained in dictionaries organized into a list with key `evidences`, which then contains `obj_surface` (the answer). Each evidence dictionary also has a `masked_sentences` list, but critically, each sentence is *not* necessarily its own question. Many of these questions do not contain "[MASK]", instead providing context for the overall question(s). While we initially yielded a concatenation of all evidence sentences, this resulted in questions that were very long (and thus used too much memory) and had multiple "[MASK]" tokens in them, so we instead just yielded the first cloze sentence containing "[MASK]" with no additional context in its own list. Note that we did *not* just yield the first sentence with "[MASK]" in it when counting TREx's questions – this is only for loading the datasets that we selected/generated. We instead counted all the sentences separately, as they are effectively each their own question.

The dataset returns a list containing the question and a string for the answer. Furthermore, as indicated above in the yield statements, we extended the PyTorch `data.IterableDataset` object to load said questions from the `.jsonl` files, as allowing indexing (as the standard `data.Dataset` object asks for) was not possible with loading all the questions into memory, which was intractable given our hardware memory constraints. This also means we could not use dataloader features like `len()` or data shuffling, the latter of which might have made training more effective or robust. Instead, the loader simply iterates through each file it is given sequentially (e.g. first outputs SQuAD, then ConceptNet, etc). Additionally, we only yielded questions where the answer is within the vocabulary space of our model. This is why Table 1 states that ConceptNet has around 23,000 questions while the file itself contains over 29,000. Finally, to prevent memory shortages, we only yielded questions with fewer than $N = 20$ words, though we likewise disabled this feature when counting the questions for Table 1

3.3 Evaluation

To evaluate the cloze-answering capabilities of our models, we used the `run_experiments.py` script from the original LAMA repository. This script feeds cloze questions from each of the aforementioned datasets into the selected MLM model, then computing the average the precision at 1 (P@1), precision at 10 (P@10), and reciprocal ranking (RR). To accommodate our experiments (as described below), we modified the script to allow for easier selection and specification of which model and dataset to load.

3.4 Training Loop

We created several modular scripts and IPython notebooks for fine-tuning BERT. Specifically, we developed it to be able to start training on any specified saved set of BERT weights, as well as on any dataset formatted in the same way as LAMA. This modularity is likewise necessary to accommodate later experiments (listed below).

We initially decided to use similar training hyperparameters as that of the original BERT paper (Devlin et al., 2019). Specifically, we used the PyTorch implementation of the Adam optimizer with learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L_2 weight decay of 0.01, and linear learning rate decay (Kingma and Ba, 2017). We chose to optimize cross entropy loss, as is standard for many categorical prediction tasks. The batch size, number of training epochs, and total amount of training data varies depending on the training experiment below, so no overarching quantities are listed here.

After testing, we found that the above hyper-

parameters were infeasible or inappropriate for our use case, often leading to poor training performance. As such, we had to empirically test a series of hyperparameters to find the best performance. We tested learning rates of $5e-5$, $3e-5$, $1e-5$, and $1e-6$, settling on the last one.

3.5 Fine-tuning BERT

BERT was initially trained on Wikipedia articles (Devlin et al., 2019), so although it is capable of answering questions in our dataset, it was not explicitly trained or optimized on the information contained within LAMA. Using the aforementioned training loop, we fine-tuned a pre-trained BERT model on (nearly) the entire LAMA dataset with the goal of "teaching" our model the full set of knowledge and relations that exist within it. This fine-tuned model was be compared against the pre-trained BERT in our subsequent experiments to gauge the value in training BERT on additional databases of facts.

Specifically, for fine-tuning, we chose to use 26753 total questions drawn from each of the four LAMA datasets (SQuAD, TREx, ConceptNet, and Google-RE). All these questions satisfy the parameters laid out in the Dataloader section (contain "[MASK]", have answers within BERT's vocabulary, and are less than 20 words long). The specifics can be found in Table 2. To create this dataset, we included the entirety of SQuAD (as it is relatively very small), ConceptNet, and Google-RE. We also included 5000 questions from TREx by selecting 5000 lines from its `.jsonl` files at random. Since each line contains more than one question, we simply chose the first one for each. We *could* have chosen fewer lines and chosen multiple questions from each, but we decided against this, since all such questions would have the same theme – we instead wanted the model to see a greater variance of question topics.

Note also that only a portion of these questions are usable (due to the filtering that the dataset object does). This is especially relevant for TREx and Google-RE, since those datasets have longer questions on average. Finally, for each of these datasets, we randomly split the questions into an 80%/20% training/validation data split. Table 2 displays the usable questions from each subset of LAMA.

We trained the pre-trained BERT model on this curated fine-tuning dataset for 5 epochs with a batch-size of 2 on our local GPU. For future fine-tuning (described below), we used early-stopping to pick the lowest-loss model weights, ultimately using the saved checkpoint after the end of the second epoch.

3.6 Updated LAMA

To probe the ability of BERT to learn new information and relationships, we created an "updated" LAMA dataset with new information that we re-trained the base (pretrained) and fine-tuned BERT models on. To create this dataset, we systematically changed answers to select cloze statements in the database, replacing the masked words with contextually similar responses. For example, for the statement "The current president is MASK", we could have replaced a response "Biden" with "Swift". To keep this within the scope of our project, we only modified statements in ConceptNet and SQuAD and re-trained on that dataset.

Specifically, we replaced the word "fun," "good," and "happy" with "boring," "bad," and "sad," respectively, as we found those to be among the 10 most common answers in ConceptNet. Thus, the updated LAMA data set (containing only ConceptNet and SQuAD) no longer contained the words "fun," "good," and "happy." Note that this caused a slight reduction in the number of viable questions in ConceptNet – this is because our replacement code did not ensure that the answers being replaced were the entire answer, so some words containing the aforementioned terms as sub-words were also affected. For example, "*fun*eral" became "*boring*eral", which is obviously not in BERT's vocabulary. As the decrease in number of viable questions was quite minimal and the result is quite amusing (see Table 2), we decided to keep this.

After creating this updated LAMA dataset, we re-trained both the pre-trained and fine-tuned BERT models on it, and then evaluated them according to the instructions in the evaluation section. The goal of this experiment was to test BERT's ability to change existing information and relationships that it has already learned, gaining insight into its ability to "un-learn" and re-learn facts. Specifically, we trained for 5 epochs with a batch-size of 4 on a Colab-provided GPU.

3.7 Randomized LAMA

To examine the level to which BERT overfits or "memorizes" facts, we created a "randomized" LAMA dataset that used to retrain the base BERT model on. To develop this "randomized" data set,

we pulled all the cloze statements and all the answers to said cloze statements in SQuAD and ConceptNet and randomly re-paired the statements and the answers. So for example "Dante was born in [MASK]" could have originally had the answer "Dante was born in cat" rather than "Dante was born in Florence" due to our randomization. There is no guarantee of the answers being semantically sensical. The majority of the words that are answers to the masked portion are nouns, so the phrases usually make grammatical sense, but that is not guaranteed either.

Additionally, note that this randomization increased the number of viable ConceptNet questions slightly (see Table 2). This is because, in the original ConceptNet dataset, there were questions that were less than 20 words long but had answers (masked words) that were not in BERT's vocabulary. Thus, they were rejected. Once randomized, however, some of these gained answers that were allowed, and so they were accepted by the dataset object.

After we created this "randomized" LAMA dataset, we re-trained the base BERT model on it. This experiment was performed to see the extent to which BERT was overfitting to the "randomized" LAMA dataset and essentially memorizing the information rather than more generalized learning.

We ran two training experiments on this dataset, both on the baseline pre-trained BERT model. First, we trained on the entire random dataset. We expect this to get low loss if BERT can easily memorize these nonsensical facts. Then, we trained on the same dataset, but with an 80%/20% training/validation random split, where we expect high validation loss, since validation involves testing the model on nonsensical facts that it has not seen before. Both were trained for 5 epochs with a batch-size of 4 on Colab.

4 Results

A summary of our results can be seen in the tables at the end of the paper. The tables show the mean precision at one (P@1), the mean precision at ten (P@10) and mean reciprocal rank (MRR) for the various different models across the various corpora used. The models include the base BERT model (BERT), fine-tuned BERT (ft BERT), the base BERT model trained on updated dataset (u BERT), the fine-tuned BERT model trained on the updated dataset (ft u BERT), and the base BERT models trained on the randomized datasets (r BERT or rv BERT).

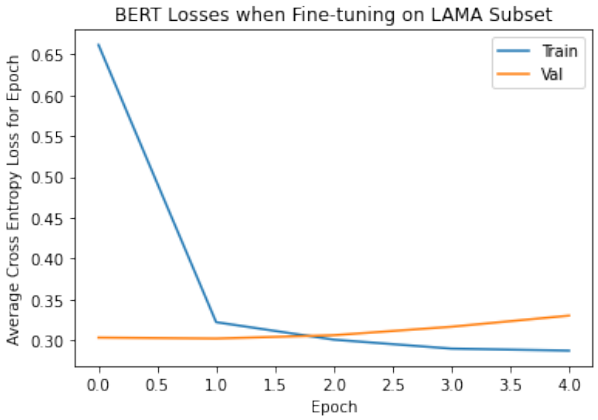

Figure 1: Training and validation loss of the pre-trained BERT model on the LAMA subset we curated for fine-tuning.

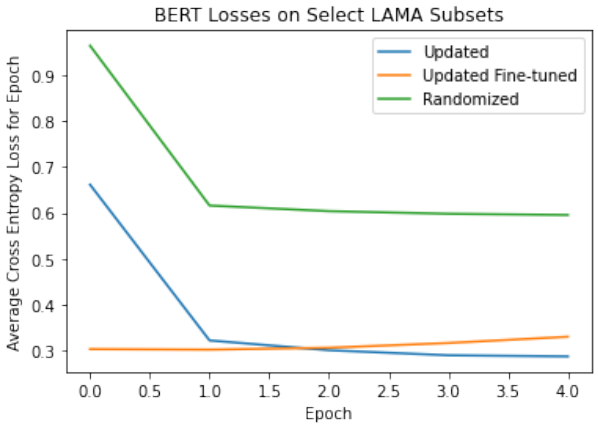

Figure 2: Training loss for BERT (baseline pre-trained and fine tuned) trained on the updated LAMA subset dataset as well as pre-trained BERT trained on the randomized dataset.

Fine-Tuning

Fine-tuning resulted in the training and validation curves shown in Fig. 1. We found that the fine-tuned model performed worse on every dataset than the pre-trained model except for ConceptNet. We assume this is because ConceptNet dominated the curated dataset used for fine-tuning. See the Discussions for further details.

Updated LAMA

When evaluating the pre-trained (*non-updated*) BERT model on the updated LAMA dataset, it achieved P@1 performances on updated (ConceptNet, SQuAD) of (0.0288, 0.136). When evaluat-

	LAMA Subset for Fine-Tuning				
	SQuAD	**TREx**	**ConceptNet**	**GoogleRE**	**Total**
Number of Questions (Train)	238	1972	16717	2467	21403
Average Words per Question (Train)	9.92	15.23	7.42	13.87	8.91
Number of Questions (Val)	64	508	4223	564	5350
Average Words per Question(Val)	9.53	15.25	7.48	13.80	8.91
Number of Questions (Total)	302	2480	20940	3031	26753
Average Words per Question (Total)	9.85	15.26	7.43	13.86	8.91
	Updated LAMA Subset				
Number of Questions	301	-	20901	-	21202
Average Words per Question	9.86	-	7.42	-	7.42
	Randomized LAMA Subset				
Number of Questions	302	-	21265	-	21567
Average Words per Question	9.85	-	7.21	-	7.25

Table 2: Various metrics for the fine-tuned, random, and updated datasets. Only viable questions (i.e. ones that are less than 20 words long and with an answer in BERT's vocabulary) were counted.

ing the updated BERT model and the fine-tuned and then updated model, they achieved P@1 performances of (0.235, 0.0662) and (0.188, .0199) respectively. We can see that both the updated and the fine-tuned and then updated models perform better than pre-trained BERT on updated ConceptNet, however fine-tuning before updating resulted in slightly inferior performance. This can be explained by the fact that the latter received conflicting information – again, see the Discussions.

Randomized LAMA

When training the pre-trained BERT model on the randomized LAMA dataset, we found that it had significantly higher losses than the other trained models (see Fig. 2). This is likely because it has to learn nonsensical mappings which would require weights significantly different from its current ones, which already encode some syntactic and semantic meaning or structure for any given word. That is, the pretrained model may have learned that "cat" is a noun, but the randomized dataset may require the model to use it as a verb to answer a randomized question, which would require large shifts in weights. Such a case may also be a one-off data point, since the other training examples are also randomized, the gradients are similarly nonsensical and unhelpful for learning to use cat for the aforementioned randomized question.

Furthermore, note that the loss we trained on is measured as the average cross entropy loss for all predicted words in a batch, *including* the ones that are given. For those words, the mapping is the identity, and so the model is able to learn this easily. When we remove these trivially-predicted words and look exclusively at the loss incurred by the masked words, the loss is an order of magnitude higher. Moreover, while we expected the loss being high for the validation set is expected (as the model never saw those examples), since both the masked-word-only loss is high (and nearly identical) for both the training **and** validation set, we conclude that there was minimal memorization of the randomized answers for the observed questions too.

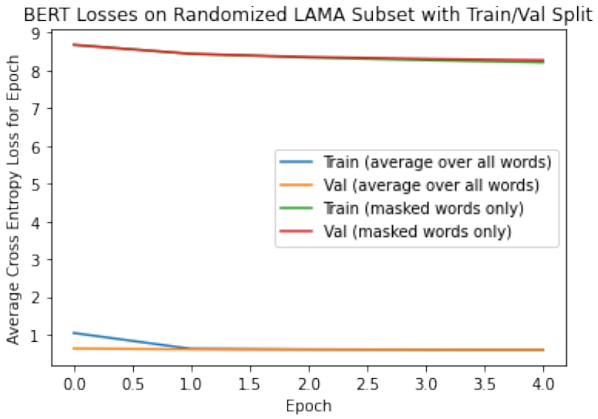

Figure 3: Training and validation loss over time when BERT is trained on the split randomized dataset. Note how, when looking just at the loss for the masked word, both the training and validation losses are very high, indicating an inability to memorize nonsense answers. Note that the loss on the training and validation sets when just looking at the masked word exclusively (green and red lines respectively) are nearly identical, and are thus hard to distinguish in the figure.

5 Discussion

We find evidence that through both updating and fine-tuning, BERT is "learning" new facts at the expense older knowledge which can be understood by analyzing three major observations in our results. The first is that the performances of the fine-tuned and randomized BERT models on our randomized ConceptNet database was better than the performance of baseline BERT. The second is that the fine-tuned BERT model performed significantly better on ConceptNet, while performing significantly worse on all of the other databases. The last is that the BERT model trained on the updated database performed better than the BERT model that was first fine-tuned and then updated. These performance trends can be seen across P@1, P@10, and MRR, so we will be referring to all of the metrics presented in Table 3 throughout this section.

Although it was trained on nonsensical sentence-word pairs, both the randomized and fine-tuned BERT models achieved better performance on the randomized ConceptNet database than baseline BERT. Although the performance should be uniform regardless of training, the models that had been trained on different versions of a majority ConceptNet database were better than a model that hadn't. This indicates that simply by virtue of training on it, BERT is capable of "learning" about the ConceptNet database distribution, without learning any of the facts in it. However, this improvement in performance is not nearly as large as the improvements between baseline BERT and the updated, fine-tuned, and updated and fine-tuned models on ConceptNet and updated ConceptNet, which indicated BERT can actually learning the specific facts in the data, and is not just improving due to learning the underlying distribution.

We can see that the original performance of BERT on ConceptNet is very poor compared to all of the other databases, indicating the knowledge initially contained within BERT did not overlap well with the distribution contained within ConceptNet. After fine-tuning on our database, which was majority ConceptNet, we find that the performance of BERT on ConceptNet significantly improved, but at the expense of performance on all other databases, despite having included samples from each of the databases. Although we updated and trained on SQuAD as well, it composed a significantly smaller fraction of our database than ConceptNet, explaining why there were no performance improvements related to it. This could indicate two things. The first is that BERT and similar models have a strict limit on the amount of knowledge they can learn, and they must "give up" older knowledge to make space for newer information. The second, which we believe to be more plausible, is that BERT overwrites knowledge that is not well represented in new distributions it is trained on, which is consistent with the general conception of how machine learning models function.

Table 3 shows that updated BERT consistently performs better than fine-tuned and updated BERT across all three metrics included (P @ 1, P @ 10, and MRR) on both data sets (updated ConceptNet and updated SQuAD). As stated in the method section, fine-tuned and updated BERT was first fine-tuned on a data set dominated by ConceptNet, with smaller amounts of SQuAD, GoogleRE, and TREx and subsequently trained on an updated data set with ConceptNet and SQuAD in which "fun","good", and "happy" were replaced with "boring", "bad", and "sad". Updated BERT performing better than fine-tuned and updated BERT seems to imply that BERT struggles to learn information after previously learning information that directly contracted it. For example, both updated BERT and updated and fine-tuned BERT were first trained on a data set that contained "Chocolate tastes [MASK]" where [MASK] is "good". In order for updated and fine-tuned BERT to learn "Chocolate is bad" (part of the updated data set), it had to unlearn the information it had learned during an earlier training stage. Meanwhile, updated BERT is being trained on a completely different knowledge distribution than what BERT was originally trained on, so there was likely minimal "overwriting". The fact that fine-tuned and updated BERT struggled with this process indicates that it does not blindly learn information and there's a certain degree to selectivity when it comes to updating what it has already learned. However, fine-tuned and updated BERT still performs better than fine-tuned BERT, indicating that although fine-tuned and updated BERT struggled to unlearn information and subsequently learn contradicting information, it still learned some of it.

Furthermore, we can observe that the difference in performance between updated BERT and fine-tuned and updated BERT is far less than the difference in performance between BERT and fine-tuned BERT on Google-RE, TREx, and SQuAD.

As discussed earlier, we suspect that the fine-tuned BERT's worse performance on these three data sets comes from the fact that the fine-tuning data set was heavily dominated by ConceptNet. In this case, fine-tuned BERT was being trained on a data set with a completely different distribution than what BERT was originally trained on. When looking at updated BERT and fine-tuned and updated BERT, the data set used to fine-tune and the data set used to update have very similar distributions, leading to less of a drop in performance. This reinforces the idea that language models have an easier time learning information with similar distributions to something they have already been trained on.

6 Conclusion

Our general goal in this project was to investigate the ability of BERT to be a knowledge source that can learn new information and change it's response to existing learned factual information. Our results indicate that BERT can function as a knowledge source and learn new facts, more so than just learning generic database statistics. However, it seems to do so at the expense of forgetting existing knowledge that isn't actively being reinforced, indicating the complete distribution of knowledge we hope to teach BERT heavily impacts its ability to learn all of it. We thus conclude that BERT and similarly structured language models can serve as knowledge bases, but they are limited in the scope of knowledge they can learn.

In terms of future avenues of work, we see three main channels to expand upon. Firstly, we were severely limited by time and hardware/computation constraints. With access to resources like more GPU memory or more balanced datasets (which would require further engineering and design), one could potentially continue the analysis we have done already, but with better- and longer-trained models that learned from more balanced datasets. Secondly, some of our initial probing techniques (like investigating to what extent newly-learned information indirectly affects other related question answers – e.g. if changing who the model predicts is President affects who it predicts is the First Lady) were not within the scope of this paper, yet are still viable areas of research. Finally, one could investigate how some of the augmentation methods (e.g. having attached memory banks) likewise might affect the analyses performed above.

Acknowledgments

We would like to thank Professor Yoon Kim for discussing our initial ideas and helping provide direction to the project.

We would also like to thank *you*, nameless reader, for taking the time to review our project.

Will would like to thank his plush wolf, Steven Derpwolf, for being a constant source of support and motivation.

References

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Hong-Kwang Jeff Kuo, Eric Fosler-Lussier, Hui Jiang, and Chin-Hui Lee. 2002. Discriminative training of language models for speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–325–I–328.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2011. Language models for machine translation: Original vs. translated texts. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 363–374, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Robert L. Logan IV, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge-graphs for fact-aware language modeling.

David Orr. 2015. Google relation extraction corpus.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Simon Razniewski, Andrew Yates, Nora Kassner, and Gerhard Weikum. 2021. Language models as or for knowledge bases.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. Conceptnet 5.5: An open multilingual graph of general knowledge.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model.

P @ 1			LM			
Corpus	BERT	ft BERT	u BERT	ft u BERT	r BERT	rv BERT
Google-RE (birth-place)	0.0160	0.00938	-	-	-	-
Google-RE (birth-date)	0.0144	0	-	-	-	-
Google-RE (death-place)	0.131	0.00784	-	-	-	-
T-REx	0.306	0.0483	-	-	-	-
ConceptNet	0.0294	0.163	-	-	-	-
SQuAD	0.135	0.0231	-	-	-	-
Updated ConceptNet	0.0288	0.155	0.235	0.188	-	-
Updated SQuAD	0.136	0.0232	0.0662	0.0199	-	-
Randomized ConceptNet	0	0.00375	-	-	0.0125	0.0113
Randomized SQuAD	0	0	-	-	0	0
P @ 10			LM			
Corpus	BERT	ft BERT	u BERT	ft u BERT	r BERT	rv BERT
Google-RE (birth-place)	0.441	0.0850	-	-	-	-
Google-RE (birth-date)	0.161	0.00188	-	-	-	-
Google-RE (death-place)	0.292	0.0876	-	-	-	-
T-REx	0.594	0.127	-	-	-	-
ConceptNet	0.129	0.430	-	-	-	-
SQuAD	0.465	0.0759	-	-	-	-
Updated ConceptNet	0.129	0.424	0.502	0.469	-	-
Updated SQuAD	0.467	0.0762	0.245	0.106	-	-
Randomized ConceptNet	0.00500	0.0188	-	-	0.0519	0.0538
Randomized SQuAD	0.00660	0.00660	-	-	0.00330	0
MRR			LM			
Corpus	BERT	ft BERT	u BERT	ft u BERT	r BERT	rv BERT
Google-RE (birth-place)	0.246	0.0374	-	-	-	-
Google-RE (birth-date)	0.0711	0.00531	-	-	-	-
Google-RE (death-place)	0.183	0.0381	-	-	-	-
T-REx	0.402	0.0675	-	-	-	-
ConceptNet	0.0647	0.254	-	-	-	-
SQuAD	0.245	0.0424	-	-	-	-
Updated ConceptNet	0.0640	0.246	0.326	0.281	-	-
Updated SQuAD	0.247	0.0425	0.128	0.0434	-	-
Randomized ConceptNet	0.00292	0.0272	-	-	0.0272	0.0266
Randomized SQuAD	0.00361	0.00303	-	-	0.00133	0.00119

Table 3: Performance metrics (P@1, P@10, and MRR) evaluating each BERT model's performance on the various datasets. The model names are BERT for the original pretrained BERT, ft BERT for fine-tuned BERT, u BERT for BERT trained on the updated dataset, ft u BERT for fine-tuned BERT trained on the updated dataset, r BERT for BERT trained on the randomized dataset, and rv BERT for BERT trained on the randomized dataset with train/validation split.