Artificial Judgement Assistance from teXt (AJAX): Applying Open Domain Question Answering to Nuclear Non-proliferation Analysis

Benjamin Wilson, Kayla Duskin, Megha Subramanian, Rustam Goychayev and Alejandro Michel Zuniga

Pacific Northwest National Laboratory
902 Battelle Blvd, Richland, WA USA
Raleigh, NC 27695
rustam.goychayev@pnnl.gov and benjamin.wilson@pnnl.gov

Abstract:

Nuclear non-proliferation analysis is complex and subjective, as the data is sparse, and examples are rare and diverse. While analysing non-proliferation data, it is often desired that the findings be completely auditable such that any claim or assertion can be sourced directly to the reference material from which it was derived. Currently this is accomplished by analysts thoroughly documenting underlying assumptions and clearly referencing details to source documents. This is a labour-intensive and time-consuming process that can be difficult to scale with geometrically increasing quantities of data. In this work, we describe an approach to leverage bi-directional language models for nuclear non-proliferation analysis. It has been shown recently that these models not only capture language syntax but also some of the relational knowledge present in the training data. We have devised a unique Salt and Pepper strategy for testing the knowledge present in the language models, while also introducing auditability function in our pipeline. We demonstrate that fine-tuning the bi-directional language models on domain specific corpus improves their ability to answer domain-specific factoid questions. Our hope is that the results presented in this paper will further the natural language processing (NLP) field by introducing the ability to audit the answers provided by the language models to bring forward the source of said knowledge.

Keywords: natural language processing, open domain question answering, bi-directional language models, nuclear proliferation detection

1. Introduction

Recently, pre-trained language model representations like Bidirectional Encoder Representations from Transformers (BERT) [1] have gained extensive attention in the NLP community and have led to impressive performance in several downstream applications. While the applications leveraging the knowledge present in the parameters of these models are growing at a rapid pace, there has also been lot of research into probing the knowledge contained in these language models. In [2], the authors demonstrate an approach of using fill-in-the-blank type statements to query the language models. The authors claim that "surprisingly strong ability of these models to recall factual knowledge without any fine-tuning demonstrates their potential as unsupervised open-domain Question Answering (QA) systems". The adoption of language models as knowledge bases has also shown several advantages; a survey [3] documenting the increasing competence of language models suggests that the language models are becoming increasingly better in tasks such as natural language understanding, questions comprehension and knowledge gap completion. Additionally, publications such as [4], [5] and [6] support the usage of BERT models specifically for QA tasks.

In this work, we are interested in leveraging the BERT model for open-domain question answering for the nuclear domain. Our focus is to develop techniques and methodologies that will help with nuclear non-proliferation analysis, which is otherwise an extremely time-consuming process. Nuclear analysts generally go through large documents of texts for specific tasks. We believe that developing tools that leverage language models for tasks such as (nuclear) domain-specific QA will greatly assist nuclear analysts.

Pre-trained language models that have been trained on articles from Wikipedia are unlikely to contain nuclear domain specific knowledge. Hence, as a first step we fine-tune these models on a domain specific corpus. Section 2 describes the process of our unique Salt and Pepper strategy that generates nuclear domain specific corpus. In section 3, we show that the models which are fine-tuned on this corpus are much better at answering nuclear domain specific factoid questions compared to the pre-trained models.

Auditability of a language model can be an important part of an analytic process, especially when it relates to data which is normally prepared by an analyst – as the analysis must point to the evidence accompanying the analytic findings. Most Machine Learning (ML) models do not contain this trail of evidence and are often referred to as "black-boxes". The basic idea of auditability is to retrieve the documents from the training corpus that contain evidence for the model's answer. Our approach to auditability is to first convert the questions and the context paragraphs into embedding vectors (a real-valued vector that represents the individual words in a predefined vector space). We experiment with approaches such as TF-IDF vectorizer [7] and Sentence BERT [8] to compute the embedding vectors. The embedding vector of the context paragraph that contains evidence for the answer will be closest to the embedding vector of the question in the vector space. Our detailed methodology of using these approaches and technical results have been summarized in the section 4 in the paper.

2. Experimental Set-Up

2.1 Data creation

We used the Stanford Question Answering Dataset (SQuAD) as the starting point for building out the dataset which would later be used in our experimentation. SQuAD is "a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable [9]." Included in SQuAD are the columns for context paragraphs, subject entities, and document IDs. This data contains over 20,000 rows which comprise the entire original SQuAD dataset. The next step is to "Salt" subject specific paragraphs by adding domain specific sentences into randomly selected subject specific paragraphs to introduce the knowledge we would later probe.

2.2 Salt: Terms in Context

The process for "Salting" starts with the creation of five lists relating to the domain specific subject. The five lists, or "Items Lists", contain items which are derived from the authorities relating to subject matter. These include:

- Nuclear Weaponization; [10]
- Nuclear Fuel Fabrication; [11]
- Nuclear Gas Centrifuge; [12]
- Methamphetamine; [13] and finally,
- Silly Stuff – our own creation of unrelated words.

These items are then randomly selected from the list and populated into another list known as "Carrier-Sentences". The sentences in the "Carrier-Sentences" list contain the subject, item, and location as fill-in-the-blank placeholder tokens. For example, one of the sentences in our "Carrier-Sentences" list is "[WHO] also provided information on the [Y] research and development activities at [WHERE]".

The [WHO] in the sentence is replaced by chosen token representing an individual from the SQuAD dataset. The [WHERE] in the sentence is replaced by chosen token representing a location from the SQuAD dataset. Finally, the [Y] in the sentence is replaced by a random item from the "Items Lists". Figure 1 provides an example of the salting process for one of the five categories. There are five different [WHO]s, [WHERE]s, and [Y]s which are created to correspond with the different domains listed above. Additionally, when compiling these sentences, we also indicate how much "Salt" to add to the SQuAD dataset for each domain.

For each specified [WHO] or [WHERE] paragraph sections within the SQuAD dataset, the "Salting" code takes each [WHO] or [WHERE] section and puts them into lists. Each section then selects a random paragraph and splits it into sentences. Then, a "Salt" sentence is inserted into a random location (between split sentences) in the paragraph and recombines the paragraphs. Again, this process occurs for each of the five subject-specific "Item Lists". Once the specified number of paragraphs is "Salted" for each list, they are normalized and recombined with the remaining SQuAD paragraphs.

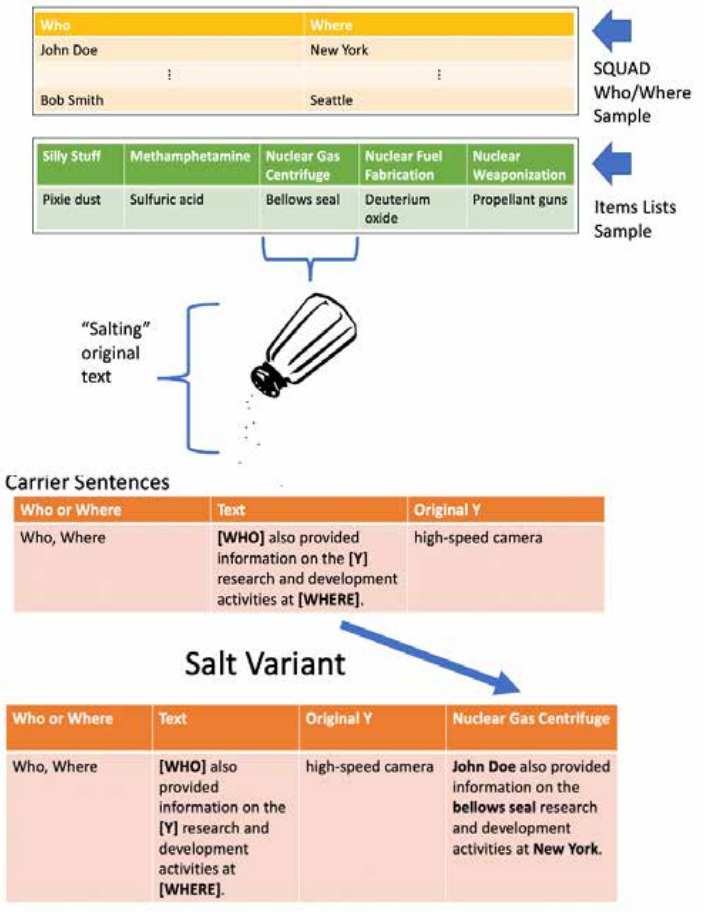

Figure 1: Flowchart depicting Salting process.

2.3 Pepper: Terms Without Context

In subsequent trials, our team decided to add meaningless sentences, or "Pepper", into the dataset to eliminate accidental knowledge recall. The "Pepper" sentences utilize the same [Y] as in the "Salted" sentences but without any mention of the [WHO] or the [WHERE].

For example, one of the meaningless sentences is "[Y] is more expensive than previously understood." Once the meaningless sentences are created, the code filters all the previously "Salted" SQuAD data and ignores the "Salted" sentences – in order to avoid "Peppering" the "Salted" sentences. The code "Peppers" the unSalted [WHO] or [WHERE] paragraphs, at random. The "Pepper" is added to eliminate any possibility that the item being mentioned in the text is being recalled when unrelated information surrounds it in proximity of the text. We could then make sure that the item is being recalled by the model based on robust knowledge retention.

2.4 Train: Domain Informed Probes and Benchmarks

Once the data is prepared, we create two model versions for our experiment: (1) a fine-tuned version of the BERT base model; and (2) a standard, pretrained BERT model. Both of these models are compared when evaluating performance of Salting technique.

Our approach for developing the fine-tuned language model involved training BERT with a batch size of 8, and dropout of 0.1; this means that for a training set consisting of about 20,000 textual examples, the model parameters are updated every 2,500 examples or so. We further initialize training to include an initial learning rate of 0.00005, following a linear learning rate schedule without weight decay. Finally, network weights were updated using Adam Optimizer. We've selected this training protocol after much trial and error, and we've found these particular settings to produce the most fruitful model for our experiments.

All pretrained models are obtained through the python HuggingFace Transformers library [14]. The fine-tuned models are trained within an Azure Databricks environment using a single GPU instance (NVIDIA Tesla V100 GPU) from an NCv3-series virtual machine. As a baseline comparison, we also evaluate the performance of the stand-alone BERT base model, without any fine-tuning.

2.5 Query: Language Model Probing

Language model probing is a way to assess the quality of the trained model by testing it against sample questions. The process for probing begins with defining a test question with a ***{tokenizer.mask_token}*** as the mask token, indicating which part of the sentence needs to be determined by the language model. The probe then looks at the ***top k*** tokens predicted by the language model for the ***{tokenizer.mask_token}*** in the test question. The value of k can range from 1 to the maximum number of tokens in the BERT vocabulary (30,522). It is often beneficial to look at more than just the top 1 token predicted by the model. In all the results presented in section 3, the value of top k is 10. These tokens are then converted/decoded into associated words using the ***tokenizer.decode*** function.

3. Results

For the purposes of evaluating our language model, we developed a set of cloze-style probe questions. Table 1 below lists some probe questions that are used to test the models. The response of the model to ***<tokenizer.mask_token>*** is treated as the predicted answer. We clearly see from Table 1 that the fine-tuned models that have been trained on domain-specific data are much better suited for domain-specific knowledge extraction. They not only provide the right answer to the probe question, but also associate those answers with a high probability.

To quantitatively assess the performance of fine-tuned language models for question answering, we performed several evaluations, which are illustrated in this section. In each of the evaluations, we considered the top 10 Recall to be

Probe Question	Pre-Trained Model Answer (Predicted Probability)	Fine-Tuned Model Answer (Predicted Probability)	Correct Answer
bellows seal is fabricated at <tokenizer. mask_token>	Mt (0.13)	Boston (0.84)	Boston
hydrogen sulphide is produced at <tokenizer. mask_token>	pH (0.06)	Detroit (0.74)	Detroit
bellows seal is developed at <tokenizer. mask_token>	Approx. (0.08)	Boston (0.95)	Boston
cylindrical rotors is located at <tokenizer. mask_token>	Approx. (0.15)	Houston (0.41)	Houston
bellows seal is owned by <tokenizer. mask_token>	Google (0.016)	Tito (0.46)	Tito
hydrogen sulphide was designed by <tokenizer. mask_token>	Siemens (0.04)	Whitehead (0.73)	Whitehead

Table 1. Probing Results of Pre-Trained and Fine-Tuned Models.

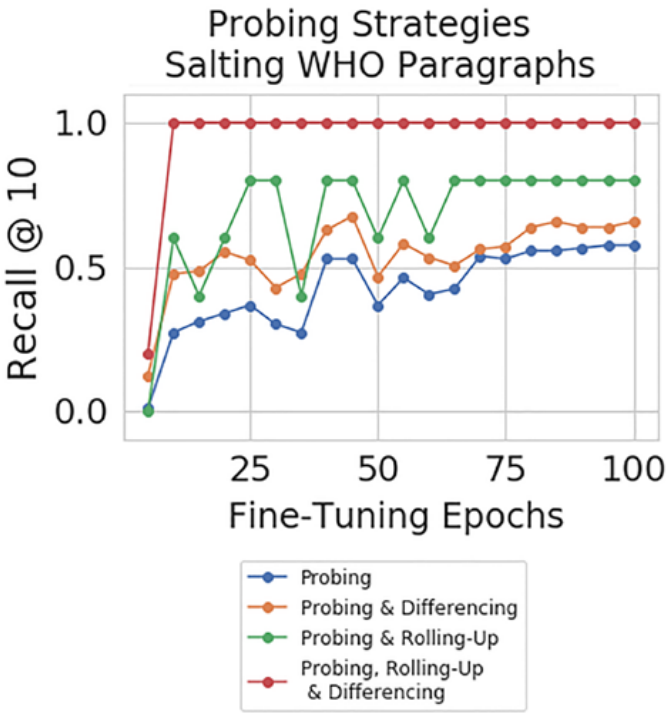

Figure 2. Effect of Probing Strategies on Performance.

the performance metric. Apart from quantitative assessment, these evaluations also shed light on several qualitative aspects of the performance of language models for questions answering, which are discussed below.

3.1 Probing Methodology

We experimented with several probing strategies for knowledge extraction from the language models. Figure 2 shows the performance of the different fine-tuned models for the different probing strategies. The blue curve which shows the least recall ability of the language models corresponds to the strategy of probing only the fine-tuned models. We observed that probing the fine-tuned models and computing the difference of the results with the pre-trained models (shown by the orange curve in the Figure 2) gives a boost to the recall metric. The best performance is shown by the red curve in the Figure which corresponds to the strategy of probing the fine-tuned models, rolling up the responses across the different probe questions and computing the difference with the pre-trained models. Overall, our results show that the probing strategy is a critical factor that influences the recall ability of the language models.

3.2 Performance Comparison on WHO and WHERE Questions

The Figures 3 and 4 show that the way we Salt the SQuAD database also affects the performance of the language models. Specifically, we find that Salting the WHO paragraphs leads to a better performance on the WHERE probe questions and vice-versa. It appears from these figures that the performance of language models as knowledge bases and the way they form semantic associations between the different tokens can be greatly influenced by the Salting strategy of the training corpus.

3.3 Performance on Salt and Pepper Data

As mentioned earlier, we also experimented with adding "Pepper" sentences (sentences that are out-of-context) to our training corpus. Figure 5 above shows the performance of the language models that are trained on this corpus. For this experiment we "Salted" and "Peppered" the WHERE paragraphs. As expected, the performance on the WHO probe questions is better. Additionally, comparing Figures 3 and 4, we see that the presence of "Pepper" sentences does not deteriorate the recall ability of the language models. This shows that the language models are robust to the presence of the confusing "Pepper" sentences in the training corpus.

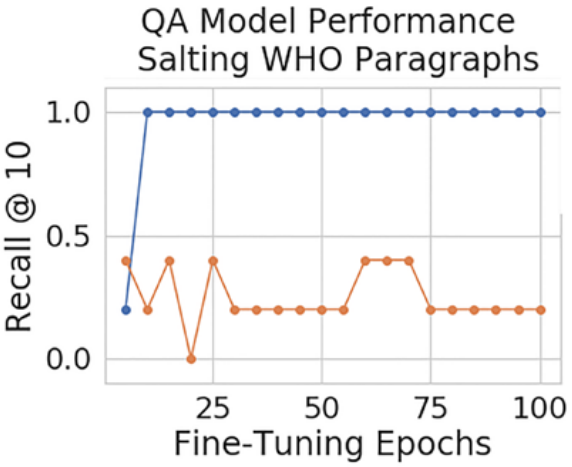

Figure 3. Model Performance – Salting WHO Paragraphs.

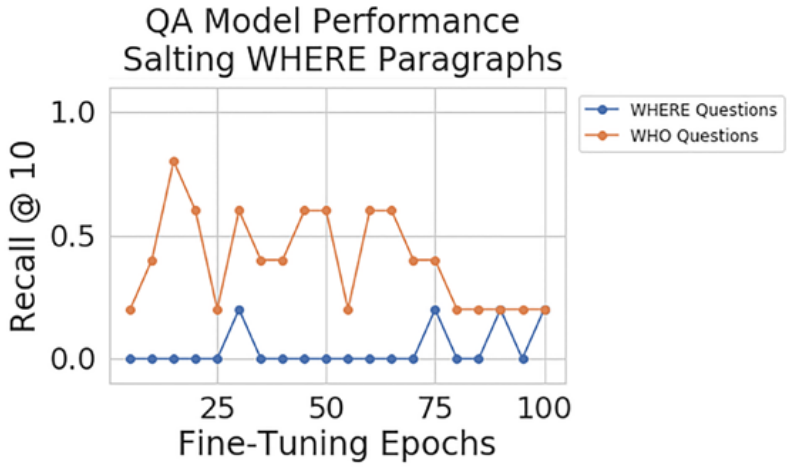

Figure 4. Model Performance – Salting WHERE Paragraphs.

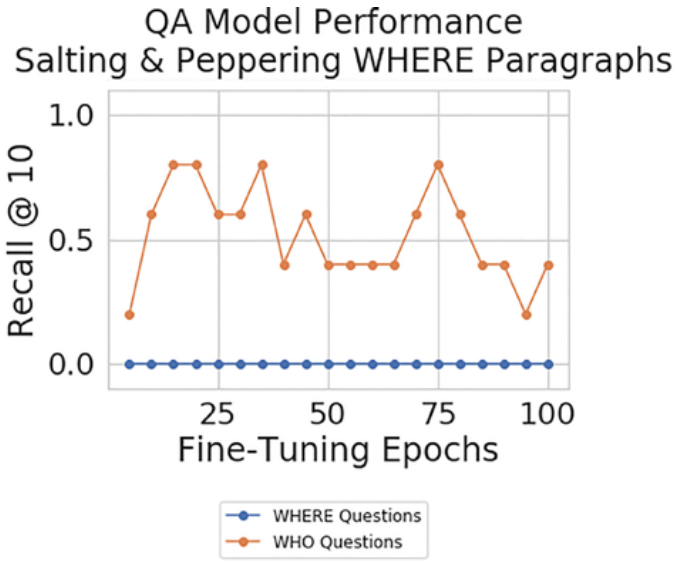

Figure 5. QA Model Performance – Salting & Peppering WHERE Paragraphs.

4. Audit

Auditability is a way to provide more insights into how the model predicted a particular answer to have an end-to-end analytical process. The basic idea of the auditability process is to look for similarities between embedding vectors of the questions and those of the contexts in the corpus. The contexts which are most similar to the questions are then retrieved. To generate the embeddings, we experimented with three techniques that are described below.

4.1 TF-IDF Vectorizer

Term Frequency — Inverse Document Frequency (TF-IDF) is a popular technique to transform textual data into meaningful numeric representation. Algorithmically, TF-IDF assigns high frequencies to those words that are more frequent in a document but not across all the documents in a corpus. For our experiments, we used the TF-IDF Vectorizer from scikit-learn library [15] to obtain the embeddings of contexts and questions. The TF-IDF Vectorizer tokenizes the documents, learns the vocabulary and inverse document weights, while also helping to encode the new documents. We use cosine similarity as a distance metric in our experiments.

4.2 BERT Embeddings

We also experimented with Transfer Learning approach by leveraging a pre-trained BERT model to obtain embeddings for both the contexts and the questions. A pre-trained BERT model provides embeddings for every token in a paragraph. We used the average of embeddings of all the tokens in the different BERT layers as a representative embedding for both context and the questions. From our experiments, we found that averaging the tokens from the 6th layer gave the best performance. These results are summarized in Table 2.

4.3 Sentence BERT Embeddings

Finally, we investigated the use of Sentence BERT architecture to obtain the embeddings for both the contexts and the questions. Sentence BERT is a modification of the off-the-shelf BERT architecture that computes semantically meaningful sentence embeddings. Of all the Sentence BERT architectures, we found that 'distilroberta-base-paraphrase-v1' gave us the best results. These results are summarized in Table 2.

4.4 Results

The first row in Table 2 below shows the auditability results on unSalted SQuAD dataset. We used development set of SQuAD database for this evaluation. Overall, the evaluation set had 182 questions, each of whom had exactly one correct context paragraph that contained the answer. The auditability task was to then retrieve the context paragraph that contained the correct answer for every question.

Additionally, the second row in Table 2 below shows the auditability results on Salted SQuAD dataset. For these evaluations, we used 85 questions and a set consisting of 160 Salted context paragraphs. For each of the 85 questions, there were 32 Salted paragraphs that contained the correct answer. The auditability task was then to retrieve one of the correct 32 Salted paragraphs for every question.

Dataset	TF-IDF	BERT tokens average across 6th layer	Sentence BERT
UnSalted SQuAD	0.91	0.74	0.91
Salted SQuAD	1.0	0.27	0.82

Table 2. Auditability metrics (Top 1 Recall).

5. Conclusion and Future Work

In this paper we demonstrated a method for testing the ability of language models to answer nuclear domain specific questions, while simultaneously introducing the auditability function in the pipeline. Our results demonstrate that language models that have been fine-tuned on domain specific corpus are much better suited for domain specific knowledge extraction compared to the pre-trained models. We have also shown that the probing methodology and the "Salting" strategy can greatly influence the ability of language models to answer domain-specific factoid questions. We have consistently observed that Salting the WHO paragraphs gives a better performance on WHERE questions and Salting the WHERE paragraphs gives a better performance on WHO questions. We think that the difference in performance is mainly due to the different Salting strategies. It appears that the way language models form semantic associations between tokens greatly depends on how we salt the corpus. In the future we would like to probe

into the multi-headed attention layers of these models to better understand this observation.

For the task of auditability, we only presented results on a subset of the corpus in this paper (Table 2). In future research, we would be interested in evaluating the auditability technique on the entire "Salted" SQuAD database. We suspect this would be a particularly challenging task for document retrieval since the entire SQuAD database consists of more than 20,000 context paragraphs. We think that further fine-tuning the Sentence BERT models on the Salted SQuAD database and then computing the embeddings for the questions and the context paragraphs will be beneficial in that case.

An opportunity that is open for future research is to leverage language models like NukeLM [16] that have been pretrained on nuclear domain data. Another area that could be further explored is the use of models like ExBERT [17] which facilitate inclusion of nuclear domain specific words in the vocabulary of the model for the task of domain specific question answering.

6. Acknowledgements

This research was supported by Laboratory Directed Research and Development Program and Mathematics for Artificial Reasoning for Scientific Discovery investment at the Pacific Northwest National Laboratory, a multiprogram national laboratory operated by Battelle for the U.S. Department of Energy under Contract DE-AC05- 76RLO1830.

7. References

[1] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." ArXiv:1810.04805 [Cs], May 24, 2019. http://arxiv.org/abs/1810.04805.

[2] Fabio Petroni and Tim Rocktäschel and Patrick S. H. Lewis and Anton Bakhtin and Yuxiang Wu and Alexander H. Miller and Sebastian Riedel (2019). Language Models as Knowledge Bases?. CoRR, abs/1909.01066.

[3] Shane Storks and Qiaozi Gao and Joyce Y. Chai, . "Commonsense Reasoning for Natural Language Understanding: A Survey of Benchmarks, Resources, and Approaches".CoRR abs/1904.01172 (2019).

[4] Soleimani, Amir, Christof Monz, and Marcel Worring. "BERT for Evidence Retrieval and Claim Verification." ArXiv:1910.02655 [Cs], October 7, 2019. http://arxiv.org/abs/1910.02655.

[5] Tushar Khot and Ashish Sabharwal and Peter Clark, . "What's Missing: A Knowledge Gap Guided Approach for Multi-hop Question Answering".CoRR abs/1909.09253 (2019).

[6] Benjamin Heinzerling and Kentaro Inui, . "Language Models as Knowledge Bases: On Entity Representations, Storage Capacity, and Paraphrased Queries". CoRR abs/2008.09036 (2020).

[7] (2011) TF–IDF. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387- 30164-8_832

[8] Reimers, Nils, and Iryna, Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." . In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2019.

[9] Pranav Rajpurkar and Robin Jia and Percy Liang (2018). Know What You Don't Know: Unanswerable Questions for SQuAD. CoRR, abs/1806.03822.

[10] infcirc254r10p2c

[11] Ibid

[12] infcirc254r13p1

[13] Methamphetamine laboratory identification and Hazards fast facts. https://www.justice.gov/archive/ndic/pubs7/7341/index.htm.

[14] 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. "Transformers: State-of-the-Art Natural Language Processing." . In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38–45). Association for Computational Linguistics, 2020.

[15] Pedregosa, F., G., Varoquaux, A., Gramfort, V., Michel, B., Thirion, O., Grisel, M., Blondel, P., Prettenhofer, R., Weiss, V., Dubourg, J., Vanderplas, D., Passos, M., Brucher, M., Perrot, and E., Duchesnay. "Scikit-learn: Machine Learning in Python".Journal of Machine Learning Research 12 (2011): 2825–2830.

[16] Lee Burke et al., "NukeLM: Pre-Trained and Fine-Tuned Language Models for the Nuclear and Energy Domains," ArXiv:2105.12192 [Cs], May 25, 2021, http://arxiv.org/abs/2105.12192.

[17] Wen Tai et al., "ExBERT: Extending Pre-Trained Models with Domain-Specific Vocabulary Under Constrained Training Resources," in Findings of the Association for Computational Linguistics: EMNLP 2020 (EMNLP-Findings 2020, Online: Association for Computational Linguistics, 2020), 1433–39, https://doi.org/10.18653/v1/2020.findings-emnlp.129.