Effect of Training Large Language Models For KG Generation on their General Language Abilities: The Cost of Specialization?

by

Sharyar Memon

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Engineering

Department of Electrical and Computer Engineering

University of Alberta

Abstract

This project focused on understanding the side effects of specializing a large language model such as GPT-2 XL on the task of Knowledge Graph Generation. It was theorized that this kind of specialization may lead to a decrease in the performance of the models in other language modelling and understanding tasks such as multiple choice selection, text entailment, word sense disambiguation and other similar tasks. In order to compare the performance of the models, a specialized model (COMET) was compared with a baseline model (GPT-2 XL) on several general language tasks. The specialized model was fully-trained on the task of Knowledge Graph generation while the baseline model was not. Both models had their base layers frozen and only the language modelling head was trained on the respective tasks. The results of the experiments indicate that the specialized model does not perform as well as the baseline model on some tasks.

Keywords: Knowledge graphs, Language Models, GPT-2 XL, COMET-DISTIL, Glue Benchmark

To my Dad

For instilling me in the confidence to ask questions and look for answers.

Programming is the closest thing we have to magic. In a fantasy story, with the right runes, a wizard can do anything. With the right codes, a programmer can do anything.

– Alex Shinsel

Contents

List of Figures

Chapter 1

Introduction

Current estimates suggest that human knowledge is doubling every 12 hours. A significant portion of that knowledge is often stored in a format that is readable by humans. However, as humans come to rely more and more on computers and their ability to interpret and understand human knowledge, so does the need for storing human knowledge in a format that is digestible by computers. This is where knowledge graphs come in. Knowledge graphs provide a way to represent and store knowledge in a graph-like structure where nodes represent entities and edges represent relationships between those entities. A prime example of a knowledge graph is the Google Knowledge Graph that is used by google to power their search engine.

Generally speaking, knowledge graphs are generated by painstaking manual human effort. This has changed with the advent of Large Language Models such as GPT-2 XL that can now be used to augment currently existing knowledge graphs[19]. This not only reduces the cost of creating a knowledge graph but also allows for much faster creation of knowledge graphs which in turn can then be used to tune downstream machine learning models[9]

1.1 Knowledge Graphs

There are multiple definitions of knowledge graphs. The most widely known one comes from Google's popularization of the word in 2012[7] where the authors imply that the knowledge that Google contains is accessible via the Google knowledge graph. Some authors describe knowledge graphs as RDF

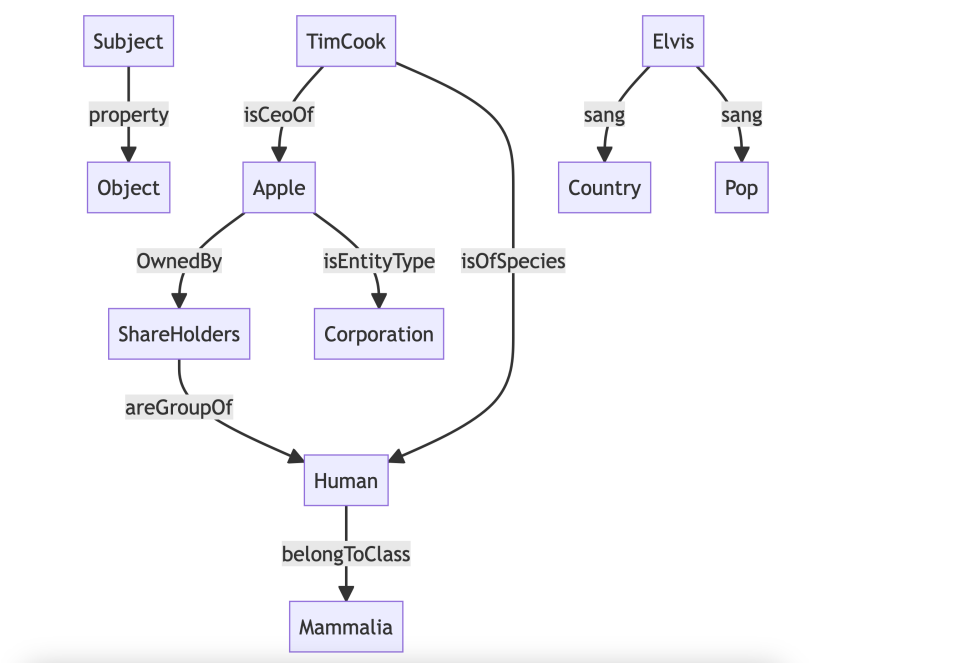

Figure 1.1: Example RDF Triples

graphs (a set of RDF triples)[3]. For our purposes, we will describe knowledge graphs as a set of RDF triples that contain a subject, a property and an object. Some example RDF triples are show in Figure1.1.

There are several types of knowledge graphs that exist today. A summary of some of the most widely used ones is available below:

1. **Domain Specific Knowledge Graphs:** These knowledge graphs are created for a specific domain. An example of this would be a knowledge graph created for medical information.

2. **Semantic Web Knowledge Graphs:** These knowledge graphs are created by extracting information from the semantic web and adhere to the RDF and OWL standards.

3. **Enterprise Knowledge Graphs:** These knowledge graphs are used to connect and collate information from different sources within an enterprise allowing for better search, recommendations and analytics.

1.2 Common Sense Knowledge Graphs

The type of Knowledge Graph relevant to this project is the Commonsense Knowledge Graph (CSKG). As the name suggests, CSKGs aim to store and represent common sense knowledge as seen from a human's perspective. This can include a wide variety of concepts that range from the simple such as "A person can be a parent of another person" to the more complex such as "A person can be a parent of another person only if the first person is older than the second". All of this these concepts help humans make sense of the world around them and are often taken for granted. However, for a computer to be able to make sense of the world around it, it needs to be able to understand these concepts. This is where CSKGs come in.

Some widely known CSKGs are ConceptNet[16] and ATOMIC[15]. The former is a knowledge graph that contains over 34 million commonsense knowledge triples that are of the form (subject, relation, object). It contains a vast array of information that has been used for a variety of Natural Language Processing tasks.

The latter is a freely available knowledge graph that contains over 877k unique commonsense knowledge triples. The triples are of the form (subject, relation, object) where the relation is one of 14 different relations (e.g.) xNeed, xIntent, xAttr etc. This knowledge graph focuses on inferential knowledge over taxonomic knowledge.

Commonsense knowledge plays a crucial role today in many machine learning applications including natural language processing and computer vision. Commonsense is often provided via a number of sources depending on the application. In order to provide a common source of common sense that can play multiple roles, Commonsense Knowledge Graphs (CSKGs) were born[6].

1.3 Large Language Models

Language models, in essence, are probability distributions over sequences of words[8]. They are used for a variety of purposes that range from the simple

such as tab completion to sophisticated text generation.

Large language models are often based on a deep learning architecture such as a transformer or a recurrent neural network. The models are trained on a large corpus of text (in the billions of documents) and require extensive computing resources to train and do inference with. The aim of the training process is to capture a generalized representation of the language that can then be used for a variety of downstream tasks.

With the advent of larger and more sophisticated language models such as GPT-3[2], the scope of usefulness for language models has expanded significantly. One such use is in the augmentation of currently existing commonsense knowledge graphs[19]. This kind of usage requires that large language models (LLMs) such as GPT-2 be trained on the task of Knowledge Graph generation. As per West *et al.*, language models fail to express common sense knowledge when prompted in a zero-shot manner. As such, the authors converted the models to COMET-DISTIL models by training them on a Knowledge Graph. We theorize that such training, while providing additional capabilities for Knowledge Graph generation, reduces other language modelling capabilities of the trained model.

1.4 Transfomers

In 2017, Vaswani et al. introduced the Transformer architecture[17]. This led to one of the widest adoptions of a single architecture in the field of Natural Language Processing. The Transformer architecture is based on the concept of self-attention. Self-attention enables a model to learn contextual relationships between words in a sentence. It does so by computing a weighted sum of the words in a sentence. The weights are computed based on the similarity between the words.

This architecture is being used in the model of interest for this project. It is also the most successful architecture for language modelling to date as evidenced in the form of BERT, GPT-3 and other LLMs.

Chapter 2

Objective of the project

The goal of this project is to understand how training (fine-tuning) a large language model such as GPT-2 XL to predict tails from a head and a relation [5] can affect its performance in other language modelling tasks when compared to its "unspecialized" version. This will help identify if this kind of training leads to a loss or a gain in capability for a language model. This is done by comparing the performance of the COMET-GPT-2 XL[19] model against the 'naive' GPT-2 XL on a variety of common modelling tasks and metrics. Additionally, this project is also a study in how to tune and use LLMs when dealing with limited compute resources as is often the case with open-source projects.

Chapter 3

Resources

3.1 Models and Tokenizers

The models used in this project come from two different sources. The GPT-2 XL model is based on the OpenAI's GPT2 model[13]. The GPT-2 XL model has 48 attention modules as compared to the 12 found in the base GPT2 model. This gives the GPT-2 XL model more capability in terms of language modelling tasks but also requires significantly more computing resources to train and work with[12]. This translates to 117 million parameters in the GPT2 model vs 1.6 billion parameters in the GPT-2 XL model. Given that the COMET-DISTIL models are based on the GPT-2 XL model, we opted to utilize the naive GPT-2 XL model as our baseline for comparison. We downloaded the GPT-2 XL model by using the Hugging Face model library[4]. As our tokenizer, we used the pre-trained tokenzier also available from the Hugging Face model library[4].

The second model we used was the COMET-DISTIL model[19] that was trained on the ATOMIC-10X knowledge graph. The COMET-DISTIL model was downloaded from the author's provided website that can be found on their GithHub page[11]. The tokenizer used with this model was also downloaded from the same source. The tokenizers for both models were adjusted to add EOS tokens to ensure that they could work with the datasets that were used for evaluating the models.

In order to train and test the models, both models had different language modelling heads (depending on the task) connected to the models. The heads

were then trained on the training set of the datasets and then evaluated on the evaluation set of the datasets.

3.2 Datasets

In order to test the performances of the respective models, several standard datasets are available. This included the widely recognized GLUE dataset[18].

The benchmark was chosen based on its wide variety of tasks and the difficulty associated with each of these tasks.

3.3 Compute Resources

To train and evaluate the performance of the respective models, Lambda Compute Instances at the University of Alberta was utilized. The machines included two Nvidia N100 Gpus with 24GB of memory. Due to the large size of the GPT-2 XL Model (5.2 Billion Parameters), the DeepSpeed[14] library from Microsoft was also utilized to help fit the models within the GPU memory.

3.4 Development Environment

1. Python 3.8.13
2. Conda 4.11.0 (Environment Manager)
3. jupyterlab 3.3.2 (IDE / Notebook)
4. PyTorch 1.11.0 (Deep Learning Framework)
5. transformers 4.27.0.dev0 (Hugging Face Library)
6. DeepSpeed 0.8.1 (Model Training Utility)

3.5 Python, JupyterLab, and Conda

We utilized a combination of tools and environments that are foundational in the modern data science and machine learning workflows: Python, JupyterLab, and Conda to conduct these experiments.

1. **Python**: Python has become the backbone of the modern Data Science and Machine Learning world due to its ease of use and the widely supported libraries built on top such as TensorFlow, Pandas and PyTorch. In addition, it is also highly performant as it is able to utilize bindings with other low level languages such as C/C++ for intensive computing tasks. This is essentially how numpy is able to do math so quickly!

2. **JupyterLab**: An interactive development environment that provides a platform for writing and executing code, visualizing data. It's an evolution of the original Jupyter notebooks.

3. **Conda**: An open-source package management system and also an environment management system. Conda makes it easy to install multiple versions of software packages and their dependencies in isolated environments. This ensures consistent and reproducible results across different stages of a project and across different team members' setups.

3.6 PyTorch

PyTorch[10] is one of the most well-known Deep Learning frameworks. Developed originally by the Facebook's Research Lab (FAIR), it is currently an integral part of the Hugging Face library which was also utilized for this project. It is used both by researchers and developers. It offers several advantages which allow us to train our models effectively even when facing low memory constraints. This is especially important when dealing with large language models such as GPT-2 XL.

Some of the offerings that come with PyTorch are it's Dynamic Computation Graph which allow us to change the model architecture on the fly. Another advantage is the rich ecosystem of libraries that either support or utilize PyTorch such as Hugging Face and DeepSpeed.

3.7 Hugging Face

Another key piece of software that was utilized to perform these experiments was the Hugging Face[20] library. It is an open source framework that provides many state-of-the-art implementations of NLP models, particularly transformer-based models such as BERT, GTP-2 and GPT-2-xl.

1. **Pre-trained Models**: The library boasts a broad selection of pre-trained models. This feature significantly eases the process of fine-tuning models for specific tasks, removing the necessity to train extensive models from the ground up. We capitalized on this by utilizing the GPT-2-xl models available from the model hub.

2. **Flexibility**: While the Transformers library seamlessly integrates with popular deep learning frameworks like PyTorch and TensorFlow, users have the freedom to select their preferred backend. In addition, Hugging Face offers native integrations with Zero and DeepSpeed libraries making model training possible even with limited hardware resources.

3. **User-friendly API**: The design of the API is intuitive, ensuring that tasks such as tokenization, training, and inference are both streamlined and accessible with minimal coding. This user-centric design approach caters to both novices and experts in the field of NLP. The library also comes with a very hand run_glue.py script that we were able to modify to run our experiments for the glue dataset.

3.8 Deepspee and ZeRO

DeepSpeed[14] is an open-source deep learning optimization library that offers several techniques for distributed training and model parallelism. One of its most renowned components is ZeRO (Zero Redundancy Optimizer), a memory optimization technology.

1. **Scalability**: DeepSpeed drastically improves the scale and speed of training. It can support model sizes of over 100 billion parameters on

current generation hardware.

2. **ZeRO (Zero Redundancy Optimizer)**: A key innovation of DeepSpeed, ZeRO optimizes the memory utilization across distributed model training. By reducing memory redundancy across the data parallelism stages, ZeRO enables the training of models that are up to 10x larger without any significant increase in computational resources.

3. **Memory Efficiency**: Through its optimization techniques, such as activation checkpointing, DeepSpeed allows for more significant model sizes and batch sizes without the need for proportional increases in memory.

4. **Integration**: DeepSpeed is designed to be easily integrated into existing PyTorch models, making it a versatile choice for many deep learning applications.

3.9 Weights & Biases

We also utilized Weights & Biases (W&B)[1], a platform designed to help teams track their machine learning experiments, visualize metrics, and share findings.

1. **Experiment Tracking**: W&B provides comprehensive logging of hyperparameters, outputs, code, and metrics. By having a centralized record, researchers and developers can easily compare different runs and iterations, making it simpler to iterate over model architectures and parameters.

2. **Visualization**: Through interactive charts and graphs, the platform facilitates a deeper understanding of model behavior. This aids in identifying patterns, anomalies, or areas of improvement in model training and validation.

3. **Reproducibility**: The platform ensures that every detail of the experiment, from code to configurations, is logged. This is crucial for both verifying results and building upon previous work without redundancy.

4. **Collaboration**: W&B promotes collaboration by allowing teams to share findings, make comments, and build on each other's work seamlessly. This feature is especially beneficial for distributed teams or projects with multiple stakeholders. This allowed our team to collaborate and share findings on this project as we were all located remotely.

For the experiments detailed in this report, Weights & Biases was employed to monitor the training process, analyze results, and ensure reproducibility across different runs.

Chapter 4

Methodology

4.1 Obtaining Models

To begin with, we obtained models from two sources. The first model was the COMET-DISTIL model which was downloaded from the original author's GitHub page[19]. The second model was the naive GPT-2 XL model. This model was obtained from the Hugging Face model hub by using their API.

Once the two models and their tokenizers were downloaded, we set up a series of experiments with different datasets. All of this was done on the University of Alberta Lambda Compute Instance via an SSH tunnel.

4.2 Preparing Datasets

To evaluate and compare the performance of our two models, it was crucial to select a benchmark that provides a comprehensive assessment across various linguistic tasks. After careful consideration, the General Language Understanding Evaluation (GLUE)[18] benchmark was chosen for the following reasons:

1. **Comprehensive Assessment**: GLUE is an aggregation of multiple data sources and tasks that span a broad range of NLP challenges. This includes tasks such as sentiment analysis, question-answering, and textual entailment, among others. Using GLUE allows for a holistic evaluation of the LLM's capabilities.

2. **Standardization**: Since its inception, GLUE has been widely adopted in the NLP community as a standard for evaluating language understanding capabilities. Comparing our models against this standard ensures that our results are in line with the broader research community's expectations and methodologies.

3. **Reproducibility and Fairness**: Given the well-defined evaluation metrics and standardized tasks, using GLUE ensures that our comparisons between the two LLM versions are fair and can be reliably reproduced by other researchers.

4. **Model Generalization**: By evaluating on GLUE, we can ascertain how well our LLM versions generalize across diverse tasks. This is crucial for understanding potential areas of improvement and the model's applicability in real-world scenarios.

5. **Community Recognition**: The results on GLUE are easily interpretable and recognizable by those familiar with NLP research. This facilitates clearer communication of our model's strengths and areas for improvement to peers and stakeholders.

Given these advantages, the GLUE benchmark provided an objective and widely-accepted means to compare and contrast the performance of our two LLM versions, ensuring both rigorous evaluation and clear communication of our findings.

We were able to download the dataset directly from the Hugging Face website. There is an accessible API that allows direct use of the datasets from the Hugging Face library within the model training code which we were able to capitalize on after.

4.3 Training Models

We tried several approaches including writing custom code to fine-tune the models for the specific tasks in the GLUE dataset. We eventually discovered

that the Transformers library comes with a handy run_glue.py script that can be found in their GitHub repository.

In order to get everything working correctly, we had to first install the Transformers library from source. Then we had to install the DeepSpeed library and try several different configurations to get the models to fit in memory and be able to train in a reasonable amount of time.

We also modified the code for the run_glue.py script as we noticed some issues while trying to collect some of the results for the GLUE benchmark dataset.

In order to conduct our study, we fine-tuned both models on the training subset for each of the GLUE benchmark sets. Then we evaluated them against the Evaluation set. Since the evaluation and the training sets are predefined, we opted to run the experiments 5 times with a different random variable set for the model training.

We then collected the predictions for the task from both models where we noticed large differences. This was done to understand if there were particular scenarios/patterns where the models did differently.

4.4 Conducting Tests and Collating Results

The training and testing of the two models took a little over 48 days to complete. Computationally, LLMs are hard to work with for this reason. We were limited in the variety of benchmarks we could run as even fine-tuning and evaluating these models without significantly more computing resources is very time-consuming.

We tracked these experiment and collected these results by using the Weights and Biases framework. The W&B report can be found via this link:

https://api.wandb.ai/report/ece-910/832fd6di

Chapter 5

Results

We created several graphs from the raw data to show how the two models perform for the various GLUE tasks.

The graphs are based on the average and standard deviations across the five runs we conducted for each of the GLUE tasks.

5.1 CoLA Results

We will start with the Corpus of Linguistic Acceptability (CoLA) results where we saw the most different between the two models.

The CoLA task is focused on a model's ability to recognize a malformed sentence (grammatically). An example of a malformed sentence that is part of the dataset is 'Who does John visit Sally because he likes'.

CoLA comprises sentences sourced from the linguistic literature, each of which is labeled in terms of its grammatically:

- **Acceptable** - The sentence is grammatically correct.
- **Unacceptable** - The sentence is grammatically incorrect.

This dataset provides a unique challenge as it demands models to capture deeper linguistic knowledge rather than relying on mere statistical patterns. Traditional metrics like accuracy are used to evaluate models on the CoLA dataset, alongside more specialized metrics like the Matthews correlation coefficient.

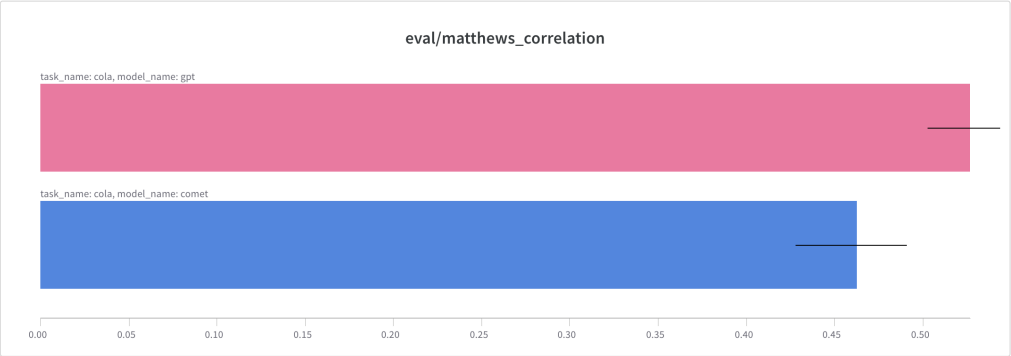

Figure 5.1: GLUE Results: CoLA

GPT-2 XL and COMET had Matthews correlation coefficient of 0.53 $\pm$ 0.02 and 0.46 $\pm$ 0.02 respectively. This was markedly the largest difference we saw between the two models for any of the GLUE tasks.

In order to understand how the models differed, we analyzed the results from the COLA task further. We used the validation subset and made predictions on it. Then we compared the predictions between the two models and the ground truth. We saw that there were 160 instances from the total subset of 1043 instances where the two models disagreed.

Upon Analyzing our results for the CoLA task, we noticed that the COMET model struggled with instances where something was happening in the past tense.

An example of this would be 'There presented itself a wonderful opportunity yesterday.' This is supposed to be a malformed sentence. The naive GPT-2 XL model is able to recognize it as such but the COMET model fails to do so.

5.2 RTE Results

The second difference we noticed between the two models was on the Recognizing Textual Entailment (RTE) task where GPT-2 XL and COMET had accuracy scores of 0.773 $\pm$ 0.009 and 0.680 $\pm$ 0.100. Given that there was a lot of variability between the runs for the COMET model, we are not conclusive about the difference in the performance between the naive GPT-2 XL and the

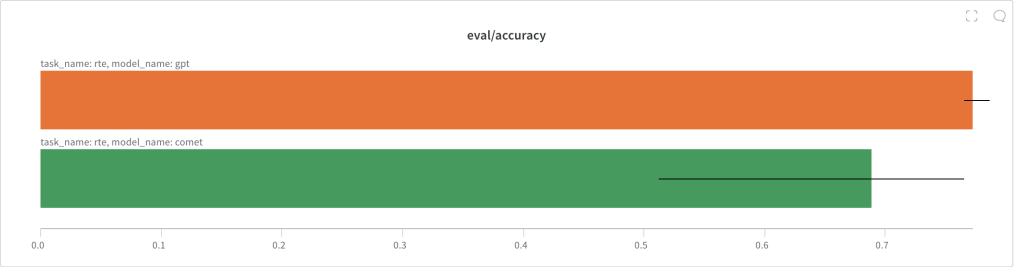

Figure 5.2: GLUE Results: RTE

specialized COMET model.

To provide some context, the RTE task is a benchmark challenge that focuses on determining whether a given piece of text (the "hypothesis") can be inferred or entailed by another piece of text (the "text" or "premise"). Typically, for any given pair of texts, the model is required to decide among three possible relations:

1. **Entailment**: The hypothesis is a logical consequence of the text. That is, if the text is true, then the hypothesis is necessarily true as well.

2. **Contradiction**: The hypothesis is logically inconsistent with the text. This means that both cannot be true at the same time.

3. **Neutral**: The relationship between the text and the hypothesis is neither one of entailment nor contradiction. The truth of the text does not guarantee the truth or falsity of the hypothesis.

For example:

- Text: "The cat is sleeping on the sofa."
  - Hypothesis 1: "There is a cat on the sofa." (Entailment)
  - Hypothesis 2: "The cat is playing outside." (Contradiction)
  - Hypothesis 3: "The sofa is blue." (Neutral)

RTE has been an important task in NLP because it captures a fundamental aspect of human language understanding.

Figure 5.3: GLUE Results: Summary

Upon analyzing the individual predictions between the models for the validation dataset, we could not find a discernible pattern such as the one we saw in CoLA. As such, we conclude that this difference in performance may just be statistical noise.

For the rest of the tasks in the GLUE dataset, we have provided a summary figure. It indicates that there was not a very large difference in the performance of the two models.

Chapter 6

Discussion

This project was focused on understanding how a Large Language Model (LLM) perform after being trained for a task such as knowledge graph generation. In order to understand the differences between the two models – a specialized model (COMET) and a baseline model (GPT-2 XL) – we compared their performances for the GLUE benchmark set and arrived at a mixed conclusion.

For the majority of the GLUE tasks, the models behave very similarly and have comparable performance. The only task where COMET model seems to have a lower performance was with the Corpus of Linguistic Acceptability task. In this case, the naive GPT-2 XL seems to perform better by a noticeable margin. Upon analyzing the predictions for this task, we noticed that COMET model seems to struggle with sentences in the past tense.

This points us in the right direction for conducting further research. Given the time constraints and the computational cost of training and testing LLMs, we had to limit this project to a set of GLUE tasks.

In the future, it would be valuable to understand how newer models such as GPT-J and GPT-3 change after they have been trained for knowledge graph generation.

References

[1] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: https://www.wandb.com/.

[2] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language Models are Few-Shot Learners," arXiv, Tech. Rep. arXiv:2005.14165, Jul. 2020, arXiv:2005.14165 [cs] type: article. DOI: 10.48550/arXiv.2005.14165. [Online]. Available: http://arxiv.org/abs/2005.14165 (visited on 06/08/2022).

[3] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger, "Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO," en, *Semantic Web*, vol. 9, no. 1, A. Zaveri, D. Kontokostas, S. Hellmann, *et al.*, Eds., pp. 77–129, Nov. 2017, ISSN: 22104968, 15700844. DOI: 10.3233/SW-170275. [Online]. Available: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-170275 (visited on 06/08/2022).

[4] *Gpt2-xl · Hugging Face*. [Online]. Available: https://huggingface.co/gpt2-xl (visited on 06/09/2022).

[5] J. D. Hwang, C. Bhagavatula, R. L. Bras, *et al.*, "COMET-ATOMIC 2020: On Symbolic and Neural Commonsense Knowledge Graphs," en, Oct. 2020. DOI: 10.48550/arXiv.2010.05953. [Online]. Available: https://arxiv.org/abs/2010.05953v2 (visited on 06/08/2022).

[6] F. Ilievski, P. Szekely, and B. Zhang, "CSKG: The CommonSense Knowledge Graph," arXiv, Tech. Rep. arXiv:2012.11490, Mar. 2021, arXiv:2012.11490 [cs] type: article. DOI: 10.48550/arXiv.2012.11490. [Online]. Available: http://arxiv.org/abs/2012.11490 (visited on 06/08/2022).

[7] *Introducing the Knowledge Graph: Things, not strings*, en-us, May 2012. [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/ (visited on 06/08/2022).

[8] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2009, ISBN: 978-0-13-187321-6.

[9] *KELM: Integrating Knowledge Graphs with Language Model Pre-training Corpora*, en, May 2021. [Online]. Available: https://ai.googleblog.com/2021/05/kelm-integrating-knowledge-graphs-with.html (visited on 07/13/2023).

[10] A. Paszke, S. Gross, F. Massa, *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703 [cs, stat], Dec. 2019. DOI: 10.48550/arXiv.1912.01703. [Online]. Available: http://arxiv.org/abs/1912.01703 (visited on 09/10/2023).

[11] peterwestai2, *Symbolic Knowledge Distillation*, original-date: 2021-10-13T02:08:39Z, Jun. 2022. [Online]. Available: https://github.com/peterwestai2/symbolic-knowledge-distillation (visited on 06/09/2022).

[12] *Pretrained models — transformers 2.2.0 documentation*. [Online]. Available: https://huggingface.co/transformers/v2.2.0/pretrained_models.html (visited on 06/09/2022).

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," en, p. 24,

[14] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, *ZeRO: Memory Optimizations Toward Training Trillion Parameter Models*, Number: arXiv:1910.02054 arXiv:1910.02054 [cs, stat], May 2020. DOI: 10.48550/arXiv.1910.02054. [Online]. Available: http://arxiv.org/abs/1910.02054 (visited on 08/05/2022).

[15] M. Sap, R. LeBras, E. Allaway, *et al.*, "ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning," en, Oct. 2018. DOI: 10.48550/arXiv.1811.00146. [Online]. Available: https://arxiv.org/abs/1811.00146v3 (visited on 06/08/2022).

[16] R. Speer, J. Chin, and C. Havasi, *Conceptnet 5.5: An open multilingual graph of general knowledge*, 2018. arXiv: 1612.03975 [cs.CL].

[17] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[18] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*, arXiv:1804.07461 [cs], Feb. 2019. DOI: 10.48550/arXiv.1804.07461. [Online]. Available: http://arxiv.org/abs/1804.07461 (visited on 09/12/2023).

[19] P. West, C. Bhagavatula, J. Hessel, *et al.*, "Symbolic Knowledge Distillation: From General Language Models to Commonsense Models," en, Oct. 2021. DOI: 10.48550/arXiv.2110.07178. [Online]. Available: https://arxiv.org/abs/2110.07178v1 (visited on 06/08/2022).

[20] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6 (visited on 06/08/2022).