Comparing Small Language Models and Large Language Models' ability to Extract Triples

Muhammad Qasim Ali

SPARC Workshop

Abstract

My research placement focus' on large language models (LLMs) and small language models (SMLs) and their ability to perform data extraction on a sample dataset. We utilise gpt-4 and a spacy model, and test them on a dataset, scoring their ability to extract entities and label predicates on a text piece. We do this using Python version 3.8, utilising prompt engineering and 'zero shot classifiers' to reduce the models' chances of error and iterate through the dataset to test the models on multiple examples. Through this testing, we were able to deduce that LLM's are superior at Zero Shot Data Extraction than SML's, however they still do fail at performing triple extraction quite frequently.

Introduction

The Introduction of Small Language Models (SLM's) and Large Language Models (LLM's) have become a widespread point of interest in the world and are becoming a key tool that is being used now and will be for the foreseeable future. These technologies have shown amazing computation, reasoning, information extraction and open domain question response skills.

Since the transformer model for machine learning, published in 2017 in the paper, 'Attention is all you need' [1] Vaswani et al.,(2017), many studies have been released comparing LLM's and SLM's in their ability preform a multitude of differing tasks, such as the ability to extract information from text. Some believe that they perform poorly at Few-Shot Information Extraction [2] Ma et al.,(2023) while others believe that they are valid option [3] Agrawal et al.,(2022). We have also preformed our own investigation into whether LLM's and SML's are good at Zero Shot Data Extraction, testing this with gpt-4 and spacy, which are LLM's and SLM's respectively, on the same set of data to compare the outputs that they give, their ability to extract triples from a piece of text and their consistency based on a knowledge graph.

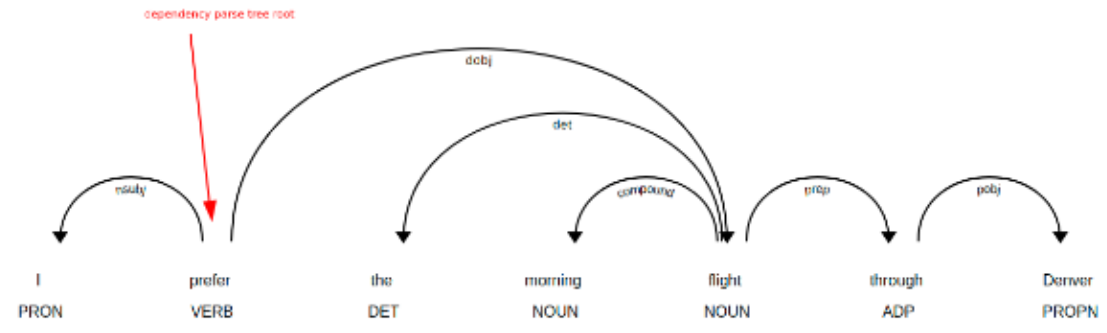

Figure 1: [4] *Transformer's self-attention mechanism simplified (2022)*

To preform said investigation, we first had to learn about a few things beforehand such as 'parts of speech'. This is the idea that every word within a sentence can be categorised as a type, such as nouns, verbs and conjunctives. This can be show in [Figure 1] with examples such as 'I' being labelled as a pronoun and 'flight' being labelled as a noun.

We also utilised the theory of ‘dependency parsing’. This is the idea that, within a sentence or text piece, words can behave as ‘roots’ for others in the sense that, for one word to make sense another must be included within the context of the sentence. For example, in [Figure 1] ‘I’ relates to ‘prefer’ as it describes an action or feeling of the person, however ‘through’ does not relate to ‘morning’ as they do not affect each other in a direct manner and intern can still make sense with its removal from the sentence.

We additionally utilised ontology, which is a set of rules or characteristics that define what something is, to define what labels we can give to entities (subjects and objects) and what predicates we can give them. We then used this ontology to set up our triples with labels and predicates that can link them.

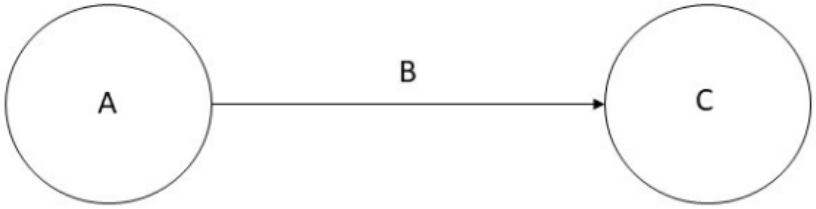

Figure 2: [5] *An introduction to knowledge graphs* (2021)

And mentioned previously and part of our end goal, we shall be trying to set up triples from the inputted text. A triple consists of two nodes’, named subject and object, as well as a predicate relating them both. For example, within the sentence “I ate a donut”, the subject is ‘I’ and object is ‘donut’, with the predicate relating the two being ‘ate’. This link can be displayed on a simple graph [Figure 2], with A being our ‘subject’, C being our ‘object’, and B being our ‘predicate’ and link between the two.

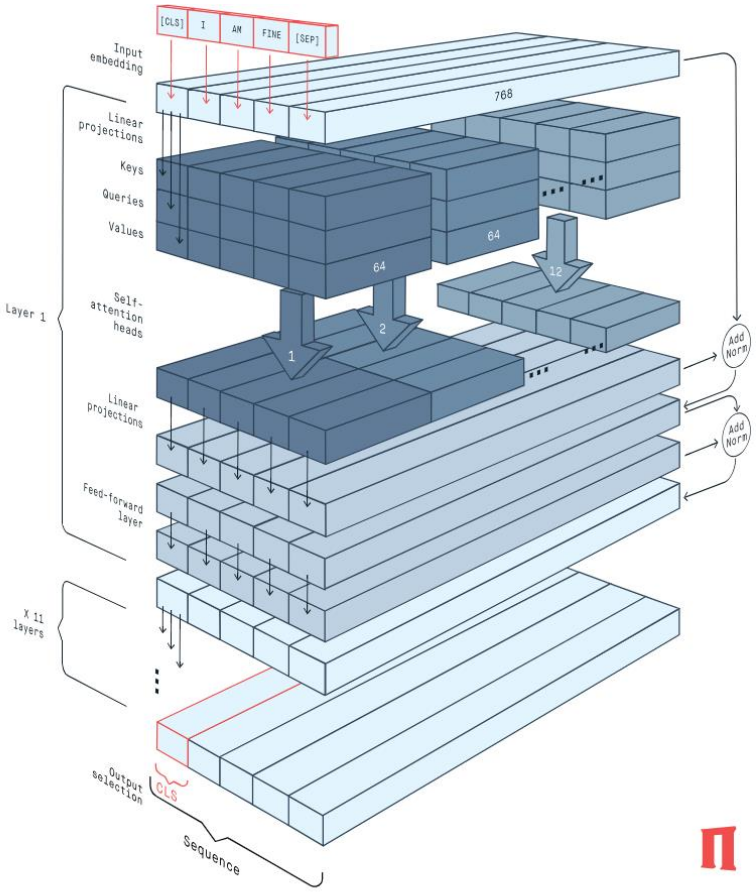

Figure 3: [11] Get ready for our pre-trained Bert, *Medium*. (2019)

The transformer model [Figure 3], which is an example for a BERT model (Bidirectional Encoder Representational from Transformers) is set up to take input text and by propagating the imbedded text, predict the next word as its output. The model preforms tokenisation on the inputted text, which is the process of dividing up a piece of text into tokens, splitting up the input statement into smaller parts that can then be worked on individually.

It then preforms word embedding on the text inputted, which is the process of expressing the semantic notation of the text as a vector, which is a list of number identifiers for the text. These can be thought to be stored in a multi-dimensional space, with words with similar semantics having similar vectors assigned.

Attention heads then run over the embedded vector, these operate akin to dependency parsing, which is examining words and their relation to one other within a sentence or phrase. This allows the model to identify which words are closely related semantically to another within the text.

It then preforms linear projection on the vectors created and assigned to the text. This process consists of multiplying the transposed vectors formed in the previous layer, which causes the array to change its formatting within the space it is held in, by the ‘weights’, which is called matrix multiplication, then adding on the ‘bias’ for that layer.

The ‘weights’ are simply the amount of influence that a node holds upon the general output of another, these weights can be positive and negative, with the higher value meaning it will have a higher influence on the final output. This causes the output to be more relevant to the text.

The ‘bias’ is then added onto the result of the previous multiplication between the vectors and ‘weights’ after being transposed. This 'bias’ is a vector, with each layer having their own vector of them.

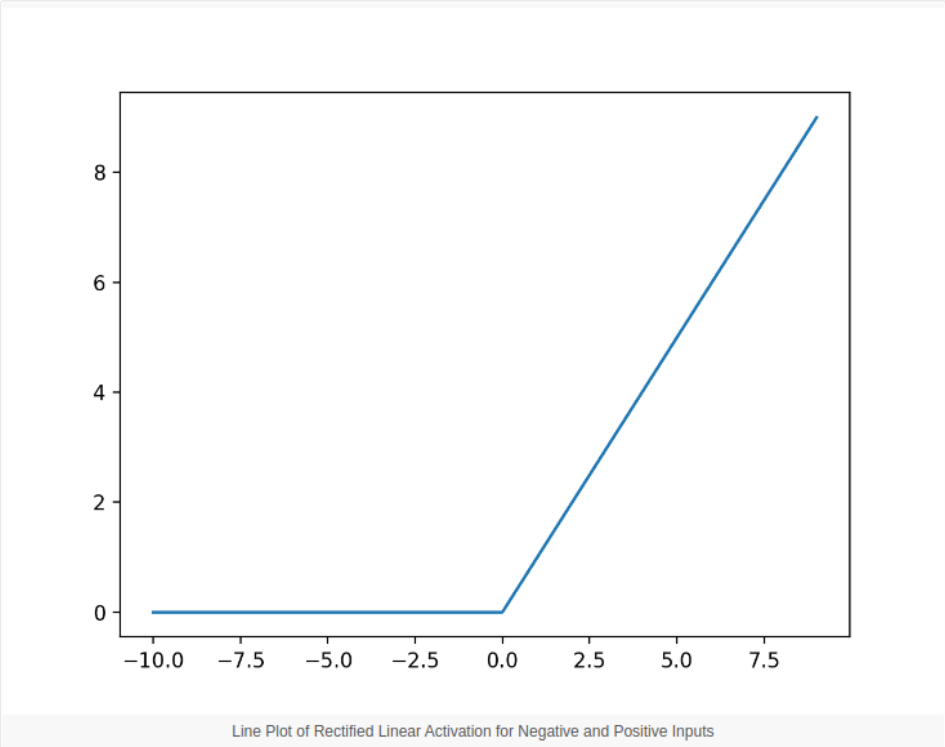

Figure 4: [6] *A gentle introduction to the rectified linear unit (ReLU), (2020)*

The model then applies relU onto this, which is a function that causes any negative values within the vector array change to zero, while also keeping any positive values the same, as shown in [Figure 4]. This is utilised as it introduces a layer of non-linearity to the model, allowing for another layer of complexity and intern getting the model to work faster and perform better.

The output of this function is once again transposed so it is formatted in the correct manner for the next layer.

To sum this process of Linear Projection algebraically, if you let P be the previous layer, W be the weights and B be bias, with ^T being the process of transposition and relu being relU then:

Next Layer = relu((P^T * W) + B^T) ^T

Then, for the add norm, the previous layer vector is added to the vector output of the Linear Layer to add more context to the model of the original text inputted into it. It also allows for layer normalisation which sorts out the distribution of intermediate layers, allowing for faster training and generalisation accuracy.

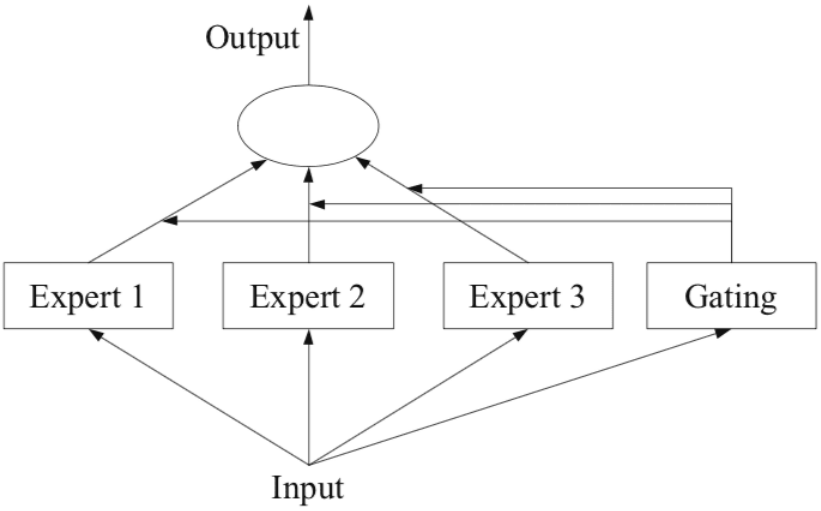

Figure 5: [7] *A gentle introduction to mixture of experts ensembles (2021)*

We can also utilise the theory of 'mixture of experts'. This is the process of creating multiple models, each trained with a different dataset so that they can specialise in a specific field, with a Gating Model managing all models within the system. The inputted data is given to each model, and they all give an output, with the Gating Model deciding the amount a model will contribute to the output dependant on their specialisation area. An example of this layout shown in [Figure 5]. This allows for a dramatic increase in model capacity while not increasing the computational tax much.

We are also going to be utilising prompt engineering, which is the process of altering the given question or prompt input to the model to try and get differing outputs that are formatted the same way. This is so that we can compare results between gpt-4 and spacy easily when utilising different prompts.

Methodology

To perform the research on knowledge extraction, we required a set of data to test the language models on their ability to extract triples from a piece of text. For this we used 'webnlg-dataset' [8] Gardent, C et al., (2017), which is a tensorflow data set compromising of multiple sets of triples on differing topics, with the information about each triple set in the form of natural language.

For example:

Text: Aaron S. Daggett fought in the Battle of Fredericksburg and received the Purple Heart.

Triples:

Subject: Aaron S. Daggett - Predicate: award - Object: Purple Heart
Subject: Aaron S. Daggett - Predicate: battle - Object: Battle of Fredericksburg

The models that we used are the spacy language model, namely 'en_web_core_trf', [9] English · spacy models' documentation (n.d.). This is the SML we used and is a roberta model trained using Onto Notes 5, ClearNLP Constituent-to-Dependency Conversion and WordNet 3.0. For our LLM we used gpt-4 [10] OpenAI, R.,(2023), created by open ai, which its training data, architecture and other aspects of the model has not been released by the company.

For the coding aspect, we utilised 'Jupyter', a python program which allows for an easy layout of code, being able to separate code into 'cells' that can be moved around, added, edited and deleted. We got our SML's and LLM's into our code by importing them. We then set up functions, which are blocks of code that only run when called, to create actions required for both models to extract information.

We then created 'zero shot classifiers' for the spacy language model. These classifiers which are the outlines to what predicates can be used to form triples from the text according to the subject and object, given to use by the ontology of the dataset we are working on, which were Politician and ComicCharacters, which has two triples per text, and CelestialBody, which has five triples per text. Additionally, we used prompt engineering on the gpt-4 input, creating a prompt input for the model that would get it to preform triple extraction, and give us the triples output for each example in the dataset according to a general format.

We then iterated through every triple we had, displaying the triples from each source against each other for every truth triple within the dataset, leaving blanks for where the language models failed to output a triple. We then scored gpt-4 and spacy against how accurate their extractions of the subject, object and predicate were compared to the truth triples, with a scoring range of zero to one.

This was done by getting the start and end token of the entity from the text given by both the truth triples and the model. We then calculated the length of the overlap between the entities. This gave us the number of characters of which both the truth triples and model triples are the same. We then divide that by the longest entity length which is value gives us the score. If they do not overlap at all then the score of the extracted entity is zero. For predicates we compare if the predicates used are the same, and if they aren't then the score is zero.

So, if one entity length was fifteen to twenty-one and the truth triple was from fourteen to nineteen then the overlap between the two would be the largest of the lower boundary and the lowest of the upper boundary, and that overlap is intern divided by the largest entity to get our score, so in this case:

Overlap = 19-15 = 4

Longest Entity = 21-15 = 6

Score = Overlap/Longest Entity

Score = 4/6

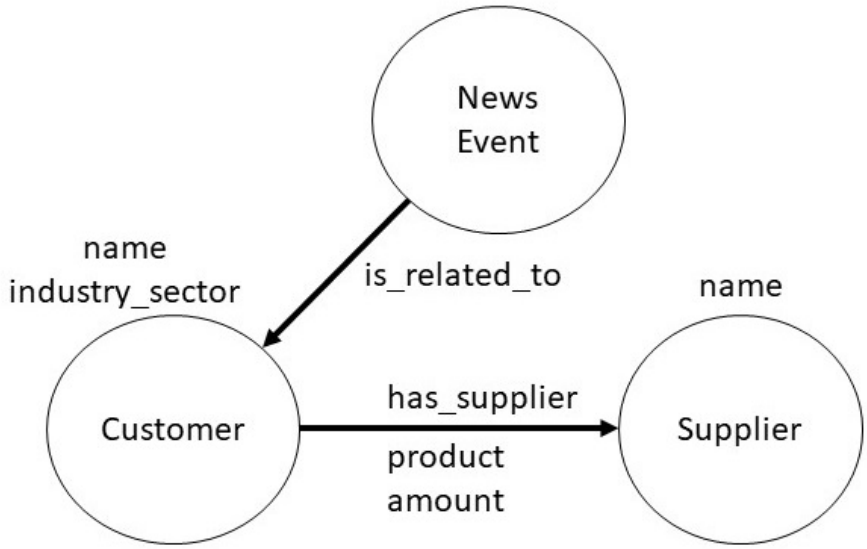

Figure 6: [5] *An introduction to knowledge graphs* (2021)

Additionally, we plotted the triples and relations between entities from all the texts on a knowledge graph for each model and the truth triples. Knowledge graphs display and organise data from multiple sources, showing information about entities in a manner that is easy to read and compare to other graphs. We can also utilise knowledge graphs to extract information in a robust manner since the way knowledge graphs are set up allows for people to easily extract information from then using graph algorithms.

Knowledge graphs consist of nodes, these are the entities of the knowledge graph, edges, which are the links between the nodes, and labels, which are a property of edges which describe the link between the nodes. An example of a knowledge graph can be seen in [Figure 6], in which 'News Event' and 'Customer' are nodes and are linked through an edge with the label 'is_related_to', however 'News Event' and 'Supplier' are not related directly, hence why there is no edge connecting the two.

Results and Discussion

Comparing results gained from testing Spacy and gpt-4 on the same set of data on their ability to perform Zero Shot Data Extraction, it seemed gpt-4 while not perfect, performed significantly better at the process of knowledge extraction than spacy.

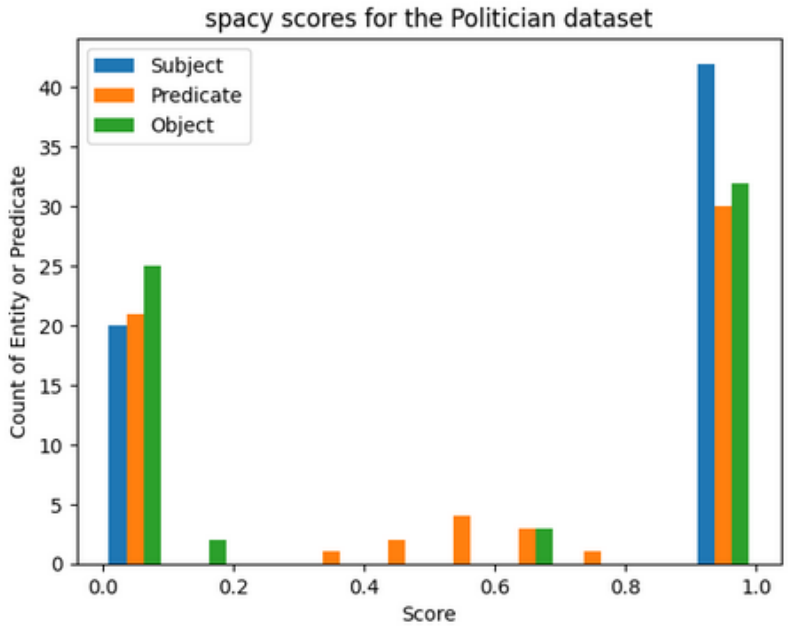

Figure 7: spacy's performance on the Politician dataset

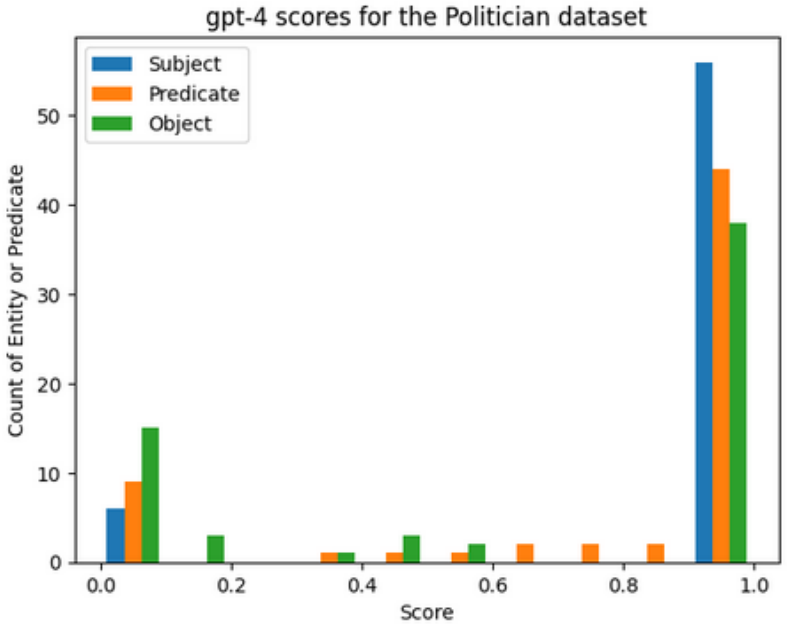

Figure 8: gpt-4's performance on the Politician dataset

On the Politician dataset, both spacy and gpt-4 performed well at data extraction however gpt-4 performed better overall, with it having higher scores on average in its extraction of entities and its labelling of predicates. This is shown in [Figure 7] and [Figure 8], in which gpt-4 received a larger number of scores above 0.5 compared to spacy. Spacy also received a score of 0 a greater number of times compared to gpt-4, meaning it failed to identify an extract an entity or extracted the incorrect entity a larger number of times than gpt-4.

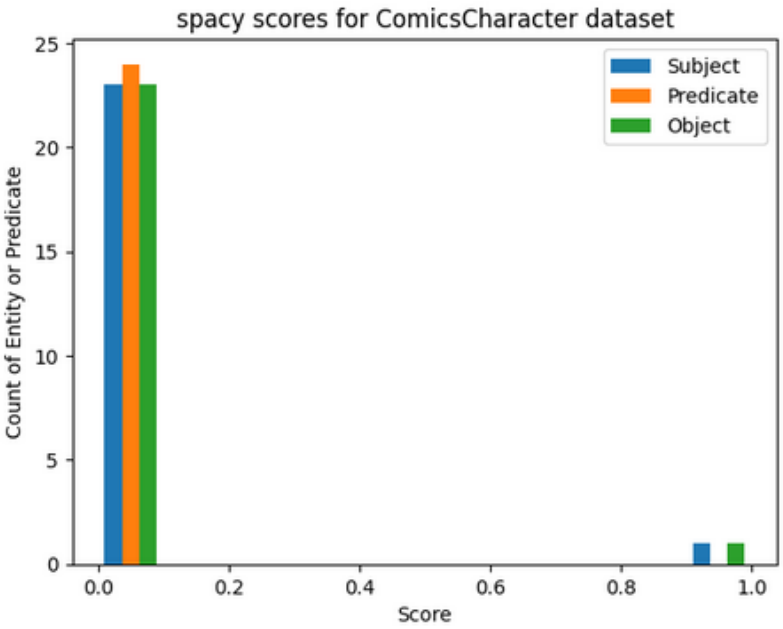

Figure 9: spacy's performance on the ComicsCharacter dataset

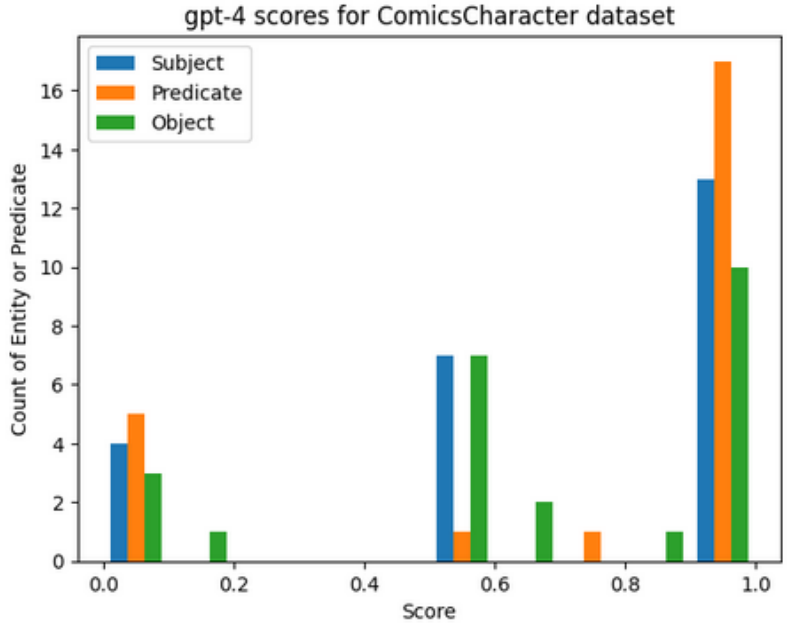

Figure 10: gpt-4's performance on the ComicsCharacter dataset

However, on the ComicsCharacter dataset, spacy preformed significantly worse than gpt-4, receiving a significantly large amount of 0 scores, showing its inability to extract information correctly from the text or at all, as seen in [Figure 9]. This completely differs for gpt-4, which succeeded in its extraction of triples a larger number of times, and while still receiving scores of 0, was able to give an output most of the time, shown by [Figure 10].

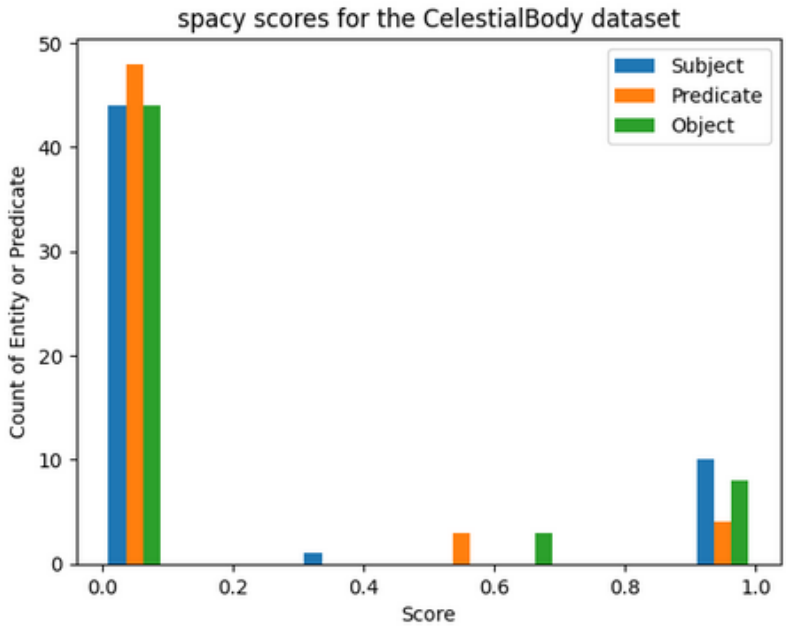

Figure 11: spacy's performance on the CelestialBody dataset

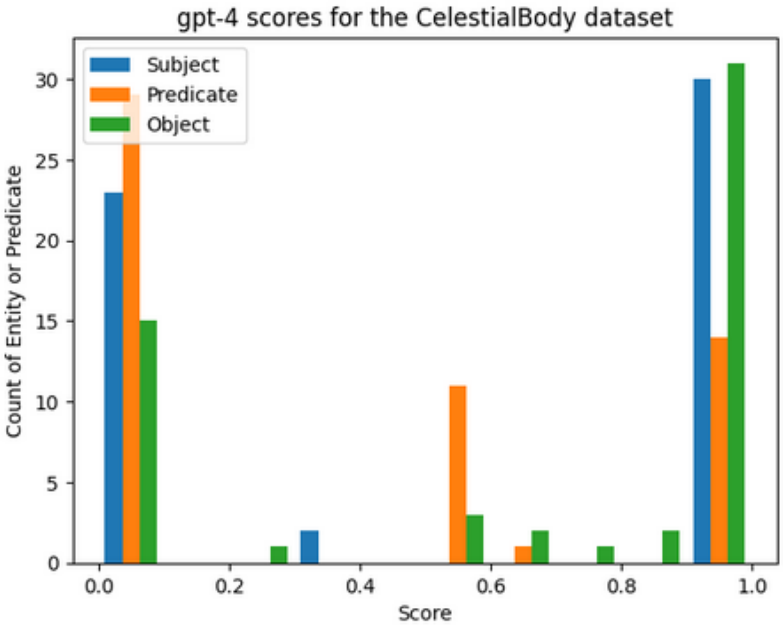

Figure 12: gpt-4's performance on the CelestialBody dataset

This result of spacy performing significantly worse than gpt-4 at triple extraction is also true for the CelestialBody dataset, which can be seen in [Figure 11] and [Figure 12] where spacy failed to extract many triples correctly, receiving mostly 0 scores and only receiving a full score a limited number of times, whereas gpt-4 received majority scores above 0.5, however it did still fail at several extractions. It also performed much worse at labelling predicates than extracting entities from the text compared to previous experiments, with the successfully labelled predicates numbering less than half of the successfully identified objects and subjects.

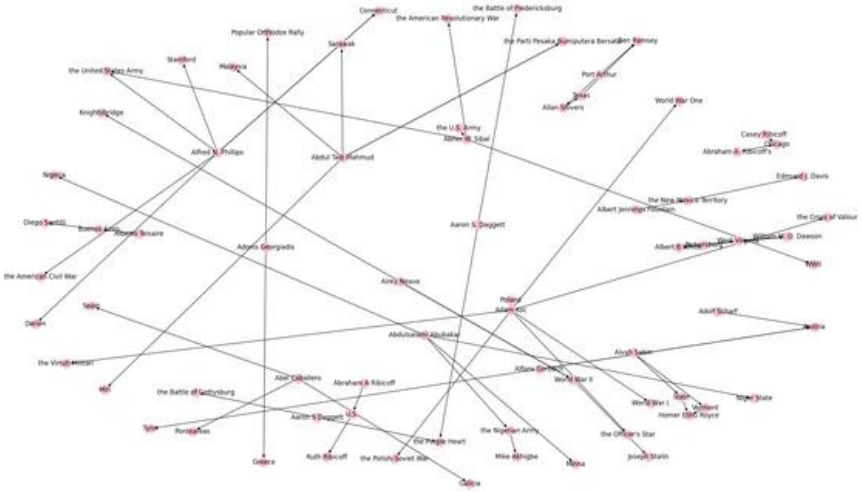

Figure 13: spacy's Knowledge Graph on the Politician dataset

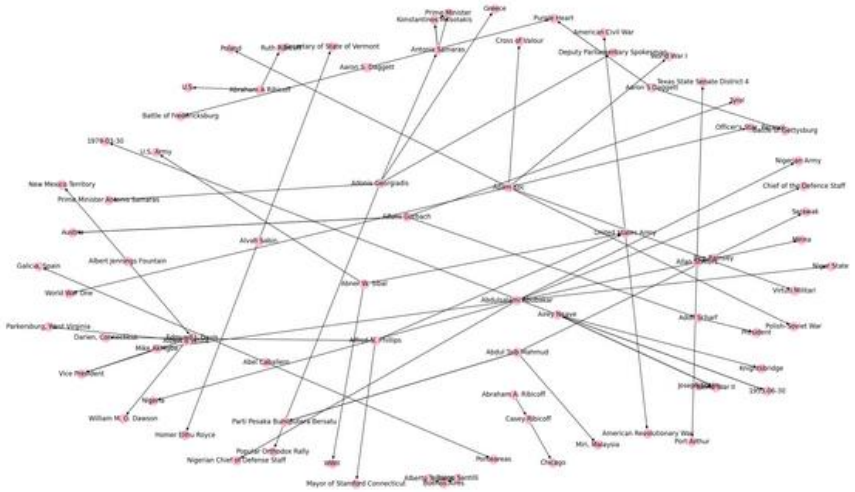

Figure 14: gpt-4's Knowledge Graph on the Politician dataset

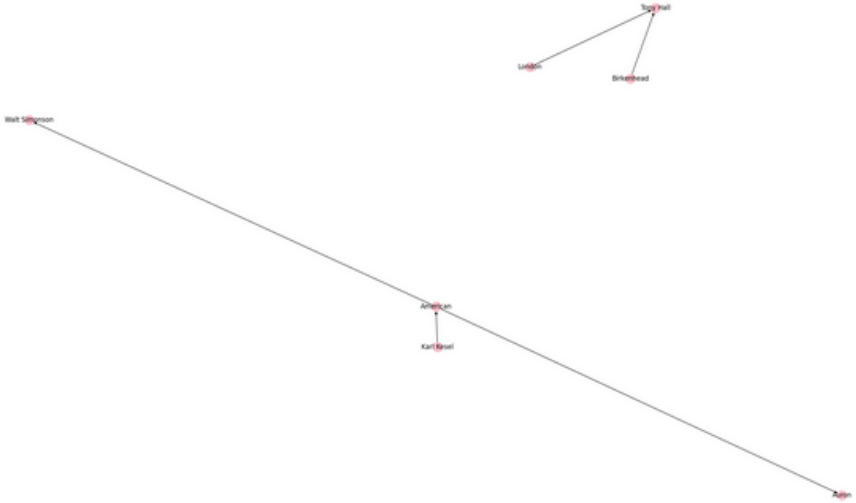

Figure 15: spacy's Knowledge Graph on the ComicsCharacter dataset

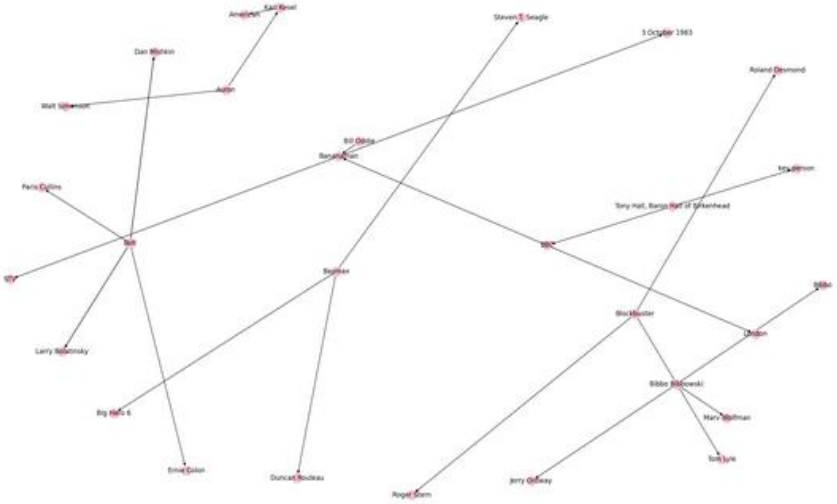

Figure 16: gpt-4's Knowledge Graph on the ComicsCharacter dataset

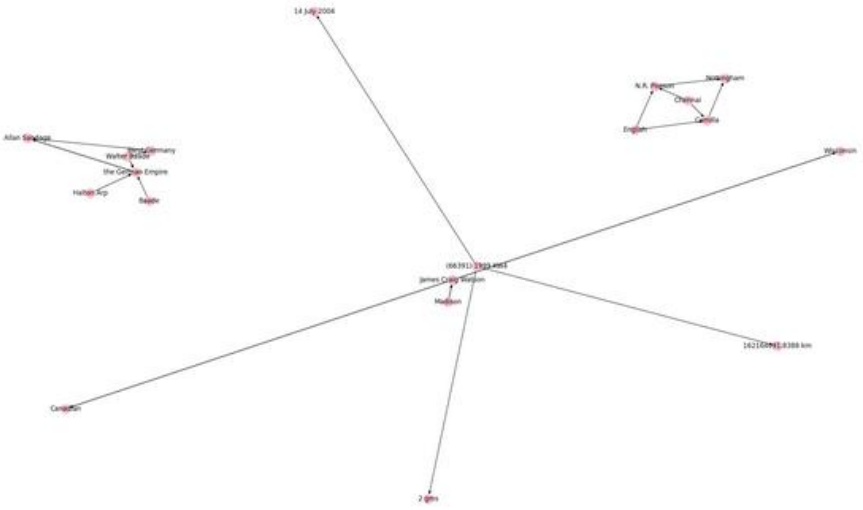

Figure 17: spacy's Knowledge Graph on the CelestialBody dataset

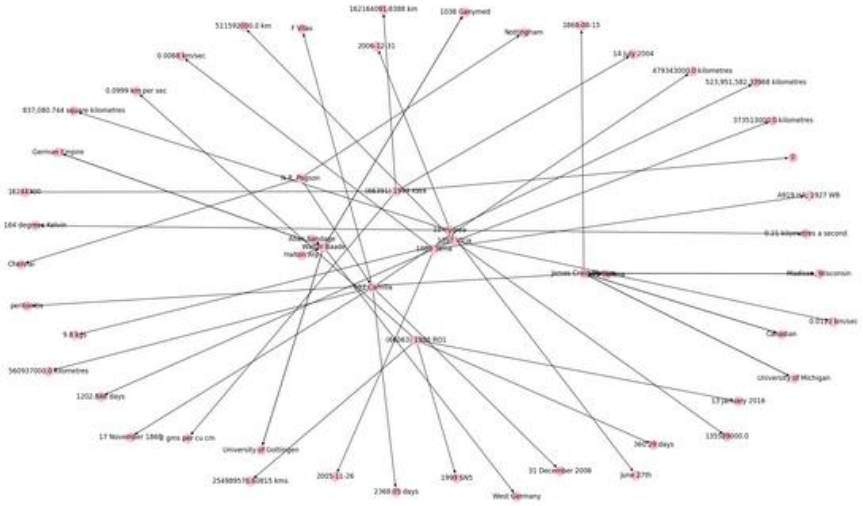

Figure 18: Spacy's Knowledge Graph on the CelestialBody dataset

Through the knowledge graphs from [Figure 13] to [Figure 18], it can also be seen that spacy preformed worse in its overall ability to extract triples compared to gpt-4. This is shown as there is less entities and links between these entities displayed on all three knowledge graphs produced from spacy's triple extraction compared to gpt-4's, meaning spacy failed to extract as many entities as possible from the text than gpt-4.

The results gained was expected based off the models we were using, and our outcome does agree with similar investigations and research on this topic. Spacy may have performed worse at data extraction than gpt-4 due to the limited data set it was trained on compared to gpt-4, which was trained on a much larger set of data. This could have led it to preform worse as it may have not seen and delt with a multitude of differing text pieces on different topics, while gpt-4 may have seen similar pieces of text to the text within the datasets, allowing it to work better as it is more accustomed to the data. Additionally, as gpt-4 is more recent in its development the data it used in its raining set is more up to date. Furthermore, as it was developed more recently than the spacy model, it is more likely to perform better due to the gap in advancements between the two models. Through our use of prompt engineering, we are also able to give gpt-4 examples of triples we want it to extract that we couldn't give to the spacy model, which may have led it to preforming better.

Conclusion

In this investigation, we tested how well SML's and LLM's can perform data extraction from a dataset. We extracted triples from a text piece with both an SLM and LLM and compared them to the original triples given by the dataset, scoring the extracted triples by how accurate they were to the originals and displayed the relationships between them on a knowledge graph, also plotting the accuracy of the results on a histogram. The experimental results show the effectiveness of LLM's and their ability to perform data extraction compared to SLM's, but also highlight the pitfalls in them both, with the LLM and SLM being unable to perform successful knowledge extraction on a consistent basis.

To continue this project, I would attempt to test both a more up to date SML against the LLM utilised in this experiment and test them on more complex datasets to see how they handle data extraction within more lengthy text pieces rather than single complex sentences, and I would also try to test the models ability to answer riddles and complex word tests that require knowledge extraction to see how it would handle a different sort of complex text. I would also attempt to test and improve extraction accuracy for both the SML and LLM, trying to continue with prompt engineering to reduce the time for the LLM to extract triples and do so with a higher success rate and create more 'zero shot classifiers' to allow for a larger range for predicate matches for the SML or even create a way for and SML to be able to perform Knowledge Extraction successfully without the requirement of the zero shot classifiers .

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.

[2] Ma, Y., Cao, Y., Hong, Y. and Sun, A., 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples!. *arXiv preprint arXiv:2303.08559*.

[3] Agrawal, M., Hegselmann, S., Lang, H., Kim, Y. and Sontag, D., 2022, December. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 1998-2022).

[4] Kosar, V. (2022) *Transformer's self-attention mechanism simplified*, *Vaclav Kosar's face photo*. Available at: https://vaclavkosar.com/ml/transformers-self-attention-mechanism-simplified (Accessed: 07 August 2023).

[5] Vinay K. Chaudhri, V., Chittar, N. and Genesereth, M. (2021) *An introduction to knowledge graphs*, *SAIL Blog*. Available at: http://ai.stanford.edu/blog/introduction-to-knowledge-graphs/ (Accessed: 07 August 2023).

[6] Brownlee, J. (2020) A gentle introduction to the rectified linear unit (ReLU), MachineLearningMastery.com. Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (Accessed: 07 August 2023).

[7] Brownlee, J. (2021) A gentle introduction to mixture of experts ensembles, MachineLearningMastery.com. Available at: https://machinelearningmastery.com/mixture-of-experts/ (Accessed: 07 August 2023).

[8] Gardent, C., Shimorina, A., Narayan, S. and Perez-Beltrachini, L., 2017, July. Creating training corpora for nlg micro-planning. In 55th annual meeting of the Association for Computational Linguistics (ACL).

[9] English · spacy models documentation (no date) English. Available at: https://spacy.io/models/en (Accessed: 14 August 2023).

[10] OpenAI, R., 2023. GPT-4 technical report. *arXiv*, pp.2303-08774.

[11] Bergenwald, H. (2019) *Get ready for our pre-trained Bert*, *Medium*. Available at: https://medium.com/peltarion/get-ready-for-our-pre-trained-bert-d653098a1496 (Accessed: 16 August 2023).

Acknowledgements

I would like to thank the Nuffield Research Placements for giving me this opportunity to take part in this investigation. I would also like to thank the hosting organisation for allowing me to investigate such an interesting topic, as well as the Reseach Placement Leader, who helped me an uncountable number of times during my placement, while also making the placement fun and relaxed. Also, thanks to the other Nuffield students who was on the placement with me, who made the placement fun and helped me during my research.