IT-UNIVERSITETET I KØBENHAVN

Bachelor Thesis

Project Code: 2849475-Spring 2020
BSc in Data Science

Improving Relation Extraction with Transformer Based Language Models and Deep Language Representations

Author
David Peter Mortensen
dmor@itu.dk

Supervisor
Manuel Rafael Ciosici
maci@itu.dk

May 15, 2020

Abstract

Relation extraction is a task in Machine Learning which seeks to identify the relationship between two or more entities, typically in the context of a sentence. In recent years relation extraction has been receiving increasing attention due to its importance in technologies such as knowledge graphs, which are used in AI-assistants and chatbots. Traditionally, Relation Extraction methods have relied on hand-designed semantic features or rule-based approaches Typically, many of these methods have struggled to capture the semantic meaning of long-range dependencies and lacks the capability of generalizing to out-of-domain text.

A new approach to relation extraction involves using a pre-trained Transformer-based Language Model as a means of pre-learning semantic features in the text, instead of relying on hand-designed semantic feature extraction. Architectures based on transformers represent a whole new level of general semantic understanding and tracking of long-range dependencies in language. This project examines how to use the OpenAI generative Pre-trained Transformer (GPT) architecture and its deep language representations in combination with additional supervised fine-tuned training as a classifier for relation extraction. We show that the fine-tuned GPT system is capable of achieving near state-of-the-art performance on the popular TACRED dataset, with an F1 score of 67.2%. Additionally, we demonstrate that the fine-tuned GPT system is capable of converging on the relation extraction task with very little training data, ultimately lowering the requirements for the scale of training data needed for the task.

1 Introduction

Relation extraction is a task in Machine Learning which seeks to identify the relationship between two or more entities, typically in the context of a sentence. Relation extraction is an essential component in popular technologies such as knowledge graphs and question answering (Yu et al., 2017). A knowledge graph is a graph database consisting of nodes and edges, where the nodes are entities and the edges their relation, as seen in Figure 1. Knowledge graphs are commonly used as the information backbone of a chat-bot or AI-assistant but have plenty of other use-cases and are, therefore, in addition to relation extraction, a field with rapidly evolving research.

Figure 1: Graph in the context of a Knowledge graph, where the nodes are entities and the edge, their relation connecting them.

Earlier work in research on relation extraction has typically consisted of rule-based systems, which primarily have searched a collection of lexicalized patterns upon the corpus (Mausam et al., 2012). As shown in (Angeli et al., 2015), rule-based systems are able to capture many cases of relational patterns found in text but lack the capability of generalization on out-of-domain text and perform poorly on relations with long-range dependencies. More recently, the task of relation extraction would be performed by machine learning classifiers that take as input some variation of transformed or encoded text corpus with two tagged entities. It would then train in a supervised manner by optimizing its weights according to a target label annotated manually or by automatic means. The target label would be the relation of the entities in the corpus; some examples from the TACRED (Zhang et al., 2017) dataset are available in Table 1. The upside of a machine learning based classifier, in contrast to a rule-based system, is the capability of more openly adapting to the various semantics in the training data, thus resulting in a system not restricted by these lexical patterns.

The traditional variation of these classifier based models relied on hand-designed semantic features extracted at a pre-processing step. However, relying on these features significantly restricts the models level of generalization and could require extensive training. Computing these features also requires additional natural language processing tools not always available in low-resource languages and requires large amounts of annotated data for relatively simple tasks. Furthermore, adding an extra step increases the complexity of the pipeline and the risk of errors. An alternative, however, is using deep language representations, the result of a language model such as the Generative Pre-trained Transformer (GPT) (Radford et al., 2018a). The deep language representations contained in such a language model are capable of learning contextualized and advanced semantic features that benefit many different downstream natural language understanding tasks, such as relation extraction. This is thanks to its transformer architecture, which offers significant complexity and efficient large-scale training due to parallelization (the transformer architecture is thoroughly explained in the Background Section 3). Ultimately, the main benefit in the context of relation extraction comes from the capability of fine-tuning the pre-trained language representations on a classification task. Extensive work on fine-tuning language models on these tasks is being explored in business and academia and is elaborately explored in the Related Work Section 2.

Extending upon the OpenAI GPT model, this paper examines how to modify and use the GPT transformer architecture and its deep language representations in combination with additional supervised fine-tuned training as a classifier for relation extraction. Since the deep language representations are unsupervised pre-trained, there is no need for additional annotated language resources or hand-designed semantic features, the semantic conceptualization is learned beforehand and thus only requires minor fine-tuning with supervised training. The python code for the system proposed in the project is open-sourced and provided online[1].

[1] https://github.com/DavidMortensen/GPT-finetuned-relation-exctraction

Example Sentence	Subject	Object	Relation
Carson, 33, has been a member of the Indianapolis City-County Council since August.	Carson	Indianapolis City-County Council	per:employee_of
Yolanda King died last May of an apparent heart attack.	Yolanda King	heart attack	per:cause of death
Jeremy and Andrew were married in Connecticut in June	Jeremy	Andrew	per:spouse

Table 1: Random examples of data points taken from the TACRED dataset.

The overall aims of the project are as follows:

- Utilize the OpenAI GPT as a Transformer based Relation Extraction model that relies on deep language representations instead of traditional hand-designed features.
- Demonstrate the benefits of pre-trained language representations in relation extraction by achieving near to state-of-the-art results on the popular TACRED dataset. Additionally, further demonstrating the capabilities of the model with various experiments.
- Compare results to what OpenAI found when fine-tuning their GPT (Radford et al., 2018a). Reproduction of the results similarly found in the work of Alt et al., who also experimented with the GPT model on the TACRED dataset.

2 Related Work

Relation Extraction Early work in relation extraction research, also sometimes referred to in academia as information extraction typically consisted of rule-based systems. Mausam et al. made use of a rule-based system that would match the input based on open pattern templates, which are relation-independent dependency parse-tree (an ordered, rooted tree structure) patterns that are automatically learned using a novel bootstrapped training set. The limitations of these rule-based systems are shown in (Angeli et al., 2015), where it is apparent how they are unable to generalize well to out-of-domain text and performs poorly on relations with long-range dependencies.

Following rule-based approaches, the common approach in optimizing relation extraction models has been providing explicit hand-designed features such as part-of-speech tags (word category tags) (Zeng et al., 2014), named entity tags (the category of each entity recognized in the text) (Song et al., 2015), and morphological features (the way in which words are formed from sub-words) (Mintz et al., 2009) as a means of programming the understanding of semantic features in the text. Initial work in the area made used statistical classifiers such as Naive Bayes or Kernel-based methods together with the use of discrete syntactic features (Zelenko et al., 2003).

In recent years, with the rise of neural networks and leaps in computational power, these methods have been succeeded by neural networks most commonly applied on a sequence of input tokens (words in a transformed list form). Sequence-based methods include Recurrent Neural Networks (RNN) (Socher et al., 2012) and Convolutional Neural Networks (CNN) (Zeng et al., 2014). Further improving on the functionality of Neural Network models, the concept of Word Embeddings (distributed representations of words) was introduced as an important milestone (Turian et al., 2010). Xu et al. integrated Shortest Dependency Path (SDP) information into an LSTM-based model, as a means of learning more robust relation representations over raw word sequences, which often suffer from irrelevant information when subjects and objects are in a long distance, achieving state-of-the-art measurements at that time on certain datasets. Zhang et al. achieved a new state-of-the-art for relation extraction on the TACRED dataset by a combination of pruning and applying graph convolutions to a dependency tree, in order to incorporate relevant information while maximally removing irrelevant content. Christopoulou et al. presented a neural graph-based model for relation

extraction that treats multiple pairs in a sentence simultaneously and considers interactions among them. The model achieved results comparable to state-of-the-art at that time on specific datasets.

Language Representations and Transfer Learning Training Deep language representations with unsupervised training has shown to be an effective way of capturing semantic information, especially handy when used as a pre-training stage of relation extraction (Howard and Ruder, 2018). Specifically, the introduction of Transformers (Vaswani et al., 2017) lead to significant advancements, with models making use of the GPT architecture, Alt et al. previously achieved state-of-the-art performance on the TACRED dataset. The Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) is another language model facilitating the transformer architecture. More recently, BERT based models have outperformed the GPT variant by making adjustments to the original architecture, chethorn2020efficient proposes a combination of a Graph Convolutional Network (GCN) and BERT in order to capture long-distance relations better and leverage local features. Finally, Baldini Soares et al. builds an agnostic relation representation from using entity-linked text in combination with BERT, which currently holds the state-of-the-art performance on the TACRED dataset.

3 Background: Language Models and The Generative Pre-trained Transformer (GPT)

Language Models can be a powerful way of constructing deep language representations for use in relation extraction. Their goal is at a high level, learning to predict the probability of a sequence of words. There exist primarily two types of different Language Models; Statistical Language Models (models that use traditional statistical techniques like N-grams and Hidden Markov Models (HMM)) and Neural Language Models (models that use different kinds of Neural Networks to model language, such as the GPT).

Figure 2: Example of how a language model predicts the next word.

A simple example in contrast to Neural Language Models are N-gram Language Models. An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language. Mathematically it asks the question: $p(w|h)$ - what is the probability of word w given a history of previous words h – where the history are $n-1$ words. We compute this probability by:

$$p(w_k|w_1, w_2, ...w_{k-1}) = p(w_k|w_{k-1}) \quad (1)$$

We approximate the history (the context) of the word w_k by looking only at the last word of the context. This assumption is called the Markov assumption. While the underlying logic behind the statistical models are relatively straight forward, this is not the case with its neural counterpart. The Generative Pre-trained Transformer (GPT) by OpenAI (Radford et al., 2018a) is at its core, an unsupervised pre-trained neural language model also capable of predicting the next word given some history. Adequately modified, the GPT model can, however, also be utilized as a semi-supervised model by fine-tuning its weights to a specific supervised learning task, such as relation extraction. The original GPT is trained on the BooksCorpus dataset (Zhu et al., 2015) containing over 7 000 unpublished books, containing a total of more than 800M words of different genres.

3.1 The Transformer

A transformer consists of two blocks, an encoder and a decoder, Figure 3 illustrates how each block of the transformer interacts with input and output in the context of a machine translation example. Machine translation is a common use-case of the transformer architecture (Wang et al., 2019) along with question answering (Lukovnikov et al., 2020).

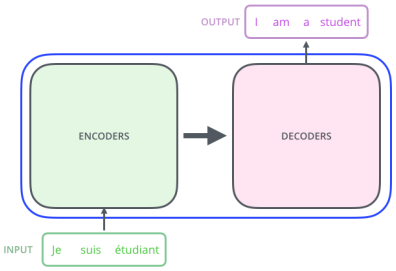

Figure 3: Machine Translation in a encoder-decoder transformer architecture. Source: (Alammar, 2019)

The technology behind the transformer is a neural network with some added complexity, and a new concept called self-attention, which is an operation performed on the input data that allows it to look at other positions in the input sequence for semantic clues that can help lead to better encoding of the word. Figure 4 shows how self-attention helps learning the dependencies between the words in a sentence. The most significant benefit of the transformer is how it implements parallelization. Unlike RNN's the transformer does not process data in sequence and thus does not need to process the beginning of a sentence before it processes the end. Due to this conceptual design, several transformer blocks can be run simultaneously, allowing for greater efficiency via parallelization. Additionally, Its self-attentive architecture allows it to capture long-range dependencies efficiently. The Transformer variant relevant in this paper, however, is the decoder-only version, which is the architecture used by the GPT.

3.2 GPT

The architecture of the GPT, as seen in Figure 5, is a 12-layer decoder-only transformer with 12 masked self-attention heads (768-dimensional states) and a position-wise feed-forward network, with 3072-dimensional inner states. The transformer decoder block essentially consists of the two multi-head self-attention and feed-forward sub-layers.

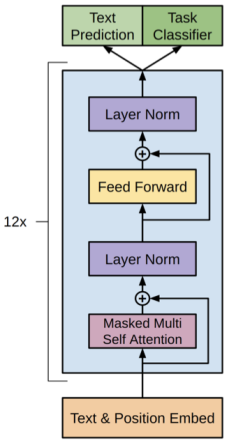

Figure 5: The Transformer architecture of the OpenAI GPT. Source: (Radford et al., 2018a)

Input Representation Before passing the input to the transformer decoder block, it is tokenized using Byte Pair Encoding (BPE) (Sennrich et al., 2016). The BPE algorithm forms a vocabulary of sub-word tokens, which is handy for dealing with rare words since the sub-word vocabulary can model new words out of its sub-words (morphology). As an example, the rare word "athazagoraphobia" would be split into the more frequent sub-words $['ath', 'az', 'agor', 'aphobia']$ with the BPE algorithm (example borrowed[2]. Beginning from single characters, the algorithm merges the most frequently co-occurring tokens into a new token and adds the new character n-gram to the vocabulary, this step is repeated until the desired number of merge operations are completed or until max vocabulary size is reached. The tokens are then represented as their byte pair encoded embeddings. An example of the iterative merge of sub-tokens can be seen in Figure 6.

[2]https://towardsdatascience.com/byte-pair-encoding-the-dark-horse-of-modern-nlp-eb36c7df4f10

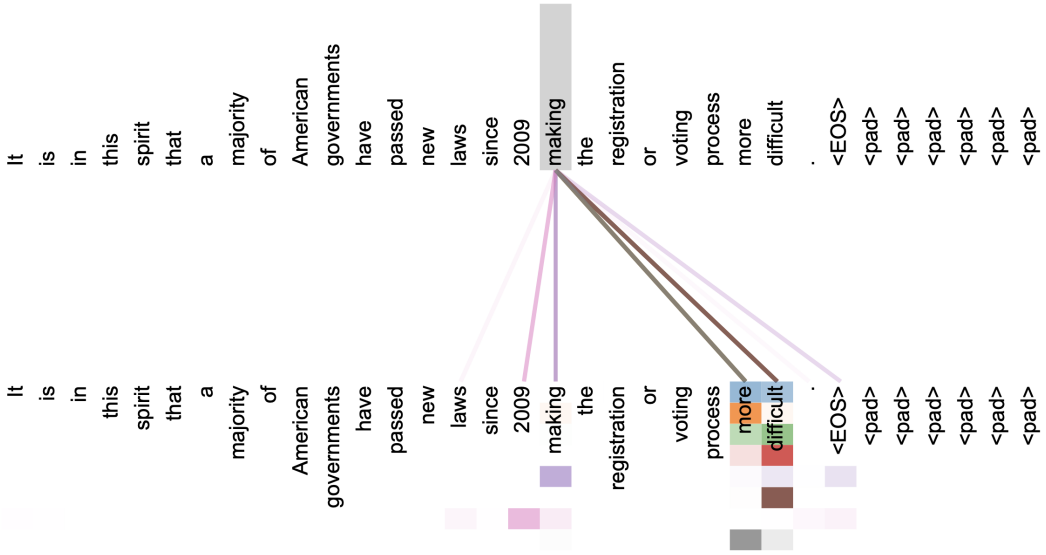

Figure 4: An illustration of how self-attention works by showing how it associates words with each other. Source: (Vaswani et al., 2017)

```
u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
u-n re-l-at-ed
un re-l-at-ed
un re-l-ated
un rel-ated
un-related
unrelated
```

Figure 6: Byte Pair Encoding of the word unrelated. Source: (Provilkov et al., 2019).

As mentioned, the transformer does not take its input in sequences but instead in parallel with no notion of position of the words. In order to counter this, Word Positional Encoding (WPE) is used, which is a signal vector representing the order of the words in a sequence; basically, a vector added to the input, as seen in Figure 7. It is generally conceived that the addition of these signal vectors to the word vectors in the embedding adds a layer of complexity and sense of position into the model.

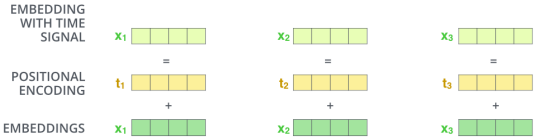

Figure 7: Word Positional Encoding (WPE) Representation. Source: (Alammar, 2019)

Masked Multi-Headed Self-Attention The embeddings are then passed into the self-attention sub-layer. The layer is responsible for looking at other positions in the input sequence when the model processes each input token, which gives the model a sense of continuous semantic understanding of the input. In Language Modelling, masked self-attention is used, which means that the operation masks (hides) the tokens ahead of the current token in time, this strategy is employed during the pre-training of the model, however, for the fine-tuning task, the model takes advantage of steering its "attention" in both directions. In the GPT, the self-attention layer is multi-headed, meaning that there are multiple attention systems (heads) deployed in parallel, this allows some attention heads to focus on long-distance dependencies, while others can focus on short-distance dependencies. Mathematically, the operation is applied by creating three vectors from each of the input token embeddings. So for each token, a Query, Key and Value vector (q, k, v) are created for each attention head, these are created by multiplying the embedding by three weight matrices (w_q, w_k, w_v), which are randomly initialized. This results in the matrices (Q, K, V), which is then used to calculate the attention and looks the following in its matrix form:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

where d_k is the dimension of the key vector k. With twelve different attention heads, we end up with as many different attention matrices

Zn. These are then concatenated into one and multiplied with an additional weight matrix W^O, which is also randomly initialized. After each sub-layer, there is a residual connection, which is the original input to the sub-layer that is then added and Normalized with Layer Normalization (Ba et al., 2016) as seen in Figure 8.

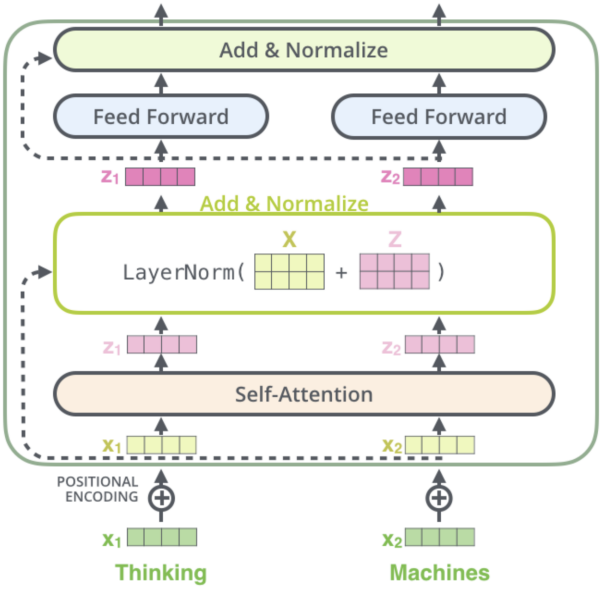

Figure 8: Residual Connection and Layer Normalization. Source: (Alammar, 2019)

Feed-Forward Layer The output is then fed into the Feed-Forward sub-layer, which is a regular neural network that takes the input and casts it to $(768 * 4)$ dimension, adds Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel, 2016) as activation function and casts it back to (768) and adds a (0.1) dropout.

Output: Language Model or Classification After the second Residual Connection and Layer Normalization block, the output is then passed to a final Linear layer of which size depends on whether the model is utilized as a language model or classification model. For the case of a Language Model, where the output represents the next word logits (probability distribution), the input is passed to a linear layer of dimension $(768, vocabularysize)$, the output is then argmaxed in order to obtain the position of the word inside the vocabulary with the highest probability. For a classification task like relation extraction, the output is also passed through a linear layer, but with size $(768, n)$, where n represents the number of classification categories. The softmax operation is then applied to the output to obtain the highest predicted category in this setup.

4 Framework

This section will describe the process of training and repurposing the GPT architecture for relation extraction. Most of the general architecture of the GPT is thoroughly explained in the Background Section 3, as such, this section will focus on the two stages of training involved in the framework. The first stage is learning a high-capacity language model on a large corpus of text. The second stage is fine-tuning the model to a relation extraction task. As it is not the first time the GPT model is repurposed as a relation extraction model, the framework of the project closely follows the work of Alt et al. and Radford et al..

4.1 Unsupervised Pre-training of Language Representations

The unsupervised pre-training of the model is based on the following mathematical methodology: Given an unsupervised corpus of tokens $\mathcal{C} = \{c_1, \ldots, c_n\}$, the language modeling objective maximizes the likelihood:

$$L_1(\mathcal{C}) = \sum_i \log P\left(c_i | c_{i-k}, \ldots, c_{i-1}; \Theta\right) \quad (3)$$

where k is the size of the context window considered for predicting the next token c_i and the conditional probability P is modeled by a neural network with the parameters Θ, which are trained using stochastic gradient descent (Robbins and Monro, 1951). The model model applies the multi-headed self-attention operation on the input, followed by position-wise feed-forward layers to compute the output probability distribution over the target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \text{transformer\_block}\left(h_{l-1}\right) \forall i \in [1, n] \\ P(u) &= \text{softmax}\left(h_n W_e^T\right) \end{aligned} \quad (4)$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the embedding matrix and W_p is the position embedding matrix.

4.2 Supervised Fine-tuning on Relation Extraction

After the pre-training with the optimization objective in Equation 3, the parameters are adapted to the supervised relation extraction task. While the model is pre-trained on plain text sentences, the relation extraction task is supervised and thus requires a structured input, i.e. a sentence and relation arguments, Figure 9 illustrates the input format.

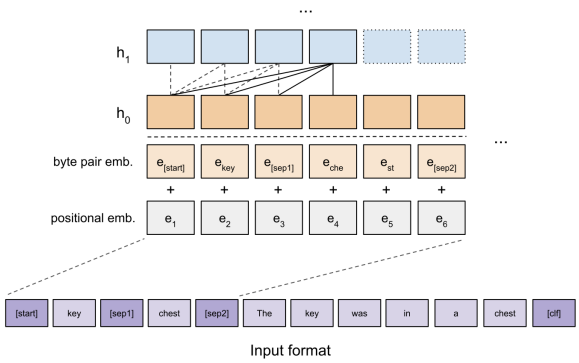

Figure 9: Structured input format for supervised fine-tuning with special delimiters, assigning specific meanings to parts of the input. Source: (Alt et al., 2019)

We assume a labeled dataset $\mathcal{D} = \left(\left[x_i^1, \ldots, x_i^m\right], a_i^1, a_i^2, r_i\right)$, where each instance consists of an input sequence of tokens $(x^1, ..., x^m)$, the positions of both relation arguments (entities) a^1 and a^2 in the sequence and a corresponding relation label r. The input is passed through the pre-trained model in order to obtain the final transformer block's activation h_L. To compute the output distribution $P(r)$ over the relation labels, a linear layer followed by a softmax is applied to the final state of the transformer block h_L^m:

$$P\left(y | x^1, \ldots, x^m\right) = \text{softmax}\left(h_l^m W_y\right) \quad (5)$$

This gives us the following objective to maximize during fine-tuning:

$$L_2(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} \log P\left(r_i | x_i^1, \ldots, x_i^m\right) \quad (6)$$

Additionally, we follow the work of (Alt et al., 2019) and (Radford et al., 2018a) by including language modelling as an auxiliary objective to the fine-tuning which has shown to lead to better generalization and faster convergence. Specifically, we optimize the following objective (with weight λ, controlling the contribution of the language model objective):

$$L(\mathcal{D}) = \lambda * L_1(\mathcal{D}) + L_2(\mathcal{D}) \quad (7)$$

5 Experiment Setup

The experiments conducted in this project are run on the supervised relation extraction dataset TACRED (Zhang et al., 2017). The results are evaluated against the similar implementation of (Alt et al., 2019) and other state-of-the-art results from external research. In regards to computational power, the different variations of models are trained on the Google Colab infrastructure with their available Tesla P100 GPU, which was powerful enough to run training in reasonable timeframes.

5.1 Dataset

TACRED: The TAC Relation Extraction Dataset (Zhang et al., 2017) contains 106 264 examples built over newswire and web text collected from the TAC KBP evaluations[3]. Examples in TACRED cover 41 relation types as used in the TAC KBP challenges (e.g., per:schools_attended and org:members) or are labeled as no_relation if no relation exists in the given example (negative examples). The class imbalance ensures that models trained on TACRED are not biased towards predicting false positives on real-world text, as this imbalance was naturally occurring. The entity mentions in TACRED are typed, meaning that they are categorized among 23 fine-grained types, such as date, location and title. The average length of the sentence examples found in TACRED are 36.4 words, reflecting the complexity of contexts in which relations occur in real-world text. The best model is selected based on the micro-averaged F1 score and is averaged over 5 runs on the test set as per convention on the dataset. The micro-averaged F1 score is calculated by using the global number of True Positives (TP), False Positives (FP) and False Negatives (FN), thus calculating micro-averaged F1:

$$microF_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (8)$$

where:

$$Recall = \frac{TP}{(TP + FN)} \quad (9)$$

and:

$$Precision = \frac{TP}{(TP + FP)} \quad (10)$$

[3]https://tac.nist.gov/2017/KBP/index.html

Using the micro-averaged F1 does not favor any classes in particular and thus weighs them equally despite any class imbalances.

Split	Number of examples
Train	68 124
Validation	22 631
Test	15 509

Total	Number of examples	Relation Types	Negative examples
TACRED	106 264	42	79.5%

Table 2: Statistics of dataset used for evaluation.

5.2 Pre-training

We reuse the language model published by Radford et al. for our experiments, which was trained on the BookCorpus dataset (Zhu et al., 2015), further explained in the Background Section 3. Pre-training of the language model is computationally extremely expensive, and it is therefore not feasible to reproduce in this setting. Additionally, the primary goal of this project is to investigate the fine-tuning capabilities only. The Language Model published by OpenAI, its training background and its architecture are described in the Background Section 3. We reuse their model's byte pair encoding vocabulary, containing 40,000 tokens. Additionally, we reuse their learned positional embeddings, which support sequence lengths of up to 512 tokens.

5.3 Model Configuration

As a guideline for setting up hyperparameters for conducting the experiments, we followed the settings reported in (Radford et al., 2018a), which in effect are relatively similar to the parameters used in the pre-training stage of the model. Therefore, we used the Adam optimization scheme (Kingma and Ba, 2014), however, with a batch size of 8 instead of 32, due to computational limitations. Alt et al. conducted several experiments with the same system architecture as ours on the TACRED dataset and previously achieved state-of-the-art results, we, therefore, try out their hyperparameter settings and further experiment with making changes to them. The best set of hyperparameters found during experiments are reported in Table 3 and compared with the ones found in (Alt et al.,

	Epochs	Learning Rate	Warm-up Learning Rate	λ
TRE	3	5.25e-5	0.002	0.5
Fine-tuned GPT	3	6.25e-5	0.002	0.5

Table 3: Best hyperparameter configuration for TACRED as found in TRE and our fine-tuned GPT variation.

2019). As the TACRED dataset contains typed entity mentions, we also introduce these types to the training input of the model. This is done by using them as an "entity masking" strategy, which replaces each entity mention by a special token, that is, either the named-entity type or the grammatical role of the entity mention, there are in total 23 of these types included in the TACRED dataset. This strategy is the common approach in state-of-the-art models on the TACRED dataset, as shown in (Zhang et al., 2018) and (Alt et al., 2019). We refer to the strategy later as "NE/GR".

6 Results

This section presents the results of our experiments with the fine-tuned GPT. The results are compared to other relevant works on the TACRED dataset, which are easily comparable. We also show that the fine-tuned GPT by far outperforms many other models without hand-designed linguistic features. We also provide results on performance with different variations of hyperparameters (benchmarked based on their validation set accuracies), amounts of training data and ablation study on the effect of unsupervised pre-training.

6.1 TACRED

On the TACRED dataset, the fine-tuned GPT model performs on par with the TRE model and achieves a micro-F1 score of 67.2 as see in Table 4, which means that we have successfully reproduced the results of Alt et al..

We observe that our model significantly outperforms many other neural systems, that does not make use of language models for their deep language representations. All the models included in the evaluation make use of the entity masking strategy explained in the Experiment Setup Section 5; in short it involves replacing each entity mention by a special token (a combination of its named entity type/grammatical role). As the TACRED dataset contains examples with high complexity and various lengths, models will need to capture the semantics of long-range dependencies, in order to achieve correct relation prediction. As seen in the results, the Language Model based systems outperform the other neural alternatives. We, however, also see that the newer variations of BERT systems (Baldini Soares et al., 2019) are able to outperform our variation of the GPT and the TRE system with a micro-F1 score of 71.5.

7 Analysis and Ablation Experiments

The results demonstrated shows that the provided model is capable of outperforming highly sophisticated systems for relation extraction and is close to state-of-the-art systems on the TACRED dataset. In this section, the process of optimizing the hyperparameters on the model are analyzed together with ablation studies on the effect of the language model pre-training and effect of training data size and number of epochs. While fine-tuned training of the model is computationally efficient compared to the pre-training stage, it still requires a powerful GPU and thus hyperparameter search methods such as grid search (where hyperparameters are tested in many or all combinations), was not feasible in this project with the available resources. The hyperparameter experiments done are, therefore variations of the best parameters found in (Alt et al., 2019). It seems rather likely that a better set of hyperparameters could exist, given that not enough research has been done. The results are reported on the pre-defined TACRED validation set, and unless differently specified, the hyperparameters are set, as the optimal set as reported in the Model Configuration Section 5.3.

7.1 Sample Efficiency and Number of Epochs

With the pre-trained deep language representations in the language model, it could be expected that relatively little training is needed in the stage 2 fine-tuning of the model since the model should already have captured most of the general seman-

Model	Precision	Recall	F1 Score
LR† (Zhang et al., 2017)	72.0	47.8	57.5
CNN† (Zhang et al., 2017)	**72.1**	50.3	59.2
Tree-LSTM† (Zhang et al., 2018)	66.0	59.2	62.4
PA-LSTM† (Zhang et al., 2018)	65.7	64.5	65.1
C-GCN† (Zhang et al., 2018)	69.9	63.3	66.4
TRE† (Alt et al., 2019)	70.1	**65.0**	67.4
Fine-tuned GPT (Ours)	70.6	64.0	67.2
BERTEM+MTB† (Baldini Soares et al., 2019)	-	-	**71.5**

Table 4: Best model performance on TACRED test set. † marks results posted in the following paper.

tic understanding. To asses the model's training efficiency in the fine-tuning stage, we sub-sample the training set and experiment with limiting the available training data to model in 10% intervals from 10% to 100%.

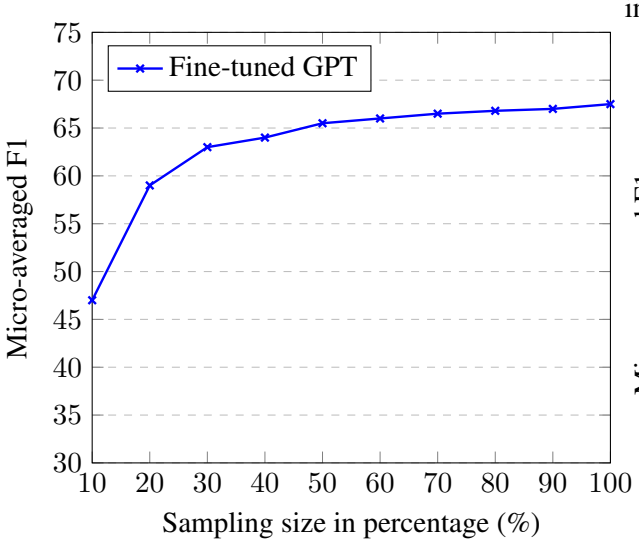

Figure 10: Micro-averaged F1 score for each sub-sample set of the training data ranging from 10% to 100%.

The results are shown in Figure 10. There is a significant performance increase in the first part of the curve as to be expected due to the learning of the weights to a specific task. The results also show that while the F1 continues to grow throughout the increase of sample size, it flattens out fairly early, meaning that a sub-sample size of only 50% achieves an impressive 65% in F1. The curve seems to stagnate at around 80% of the training data, which could indicate that this is the soft-limit for improving the model's performance with additional training data for the given task.

In regards to the number of training iterations for the model to perform in the supervised fine-tuning, we have the number of epochs. Generally, the sweet spot in the number of epochs lies where the model achieves the highest accuracy and best generalization, to the unseen data, which in this case would be the validation set.

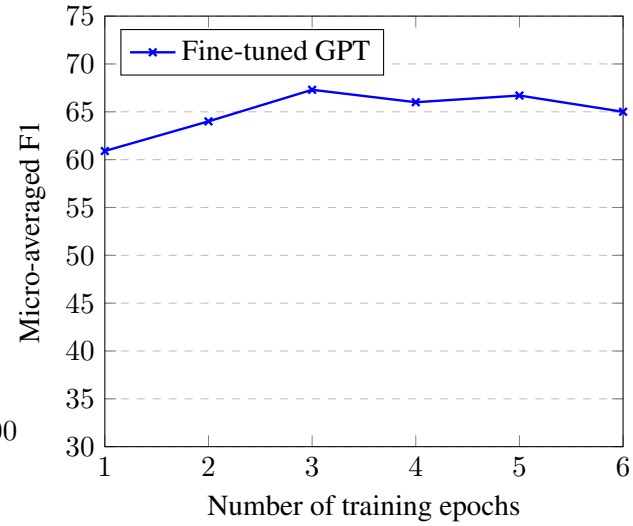

Figure 11: Micro-averaged F1 score for each epoch during fine-tuned training of the model.

The results are shown in Figure 11. We see that the highest F1 score is achieved with three epochs, as further training beyond the third epoch does not seem to improve the accuracy of the model. This further confirms the fact that the model seems to converge on the training data fairly quickly and thus does not need to run many iterations of training.

7.2 Auxiliary Language Model Mixture

As mentioned in Section 4.2, the language model is introduced as an auxiliary objective during the fine-tuning of the model, with the coefficient to control its level of contribution. Arguably, the co-

efficient could be seen as a form of adding generalization to the supervised training, since the objective in the language model would be much more generalized. However, we do not want to outweight the relation extraction optimization by too much, thus decreasing overall performance.

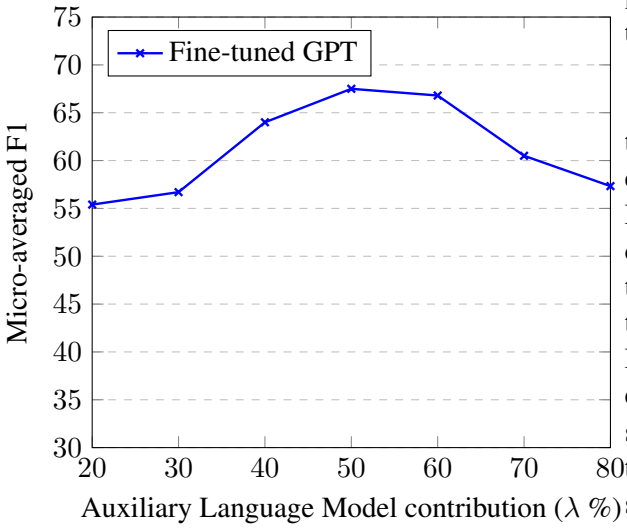

Figure 12: Micro-averaged F1 score of the different Contribution settings of the auxiliary language model objective during fine-tuning.

The results of the experiment are shown in Figure 12. It shows that the sweet spot for the language model objective is around 50%, as the F1 score maximizes at this level. We see that the F1 score significantly decreases when the coefficient is either too low or too high, this is likely due to model not adapting to the supervised task with a high λ setting, whereas with a low setting the model seems to overfit and lack in generalization.

7.3 Effect of Pre-Training

To asses the impact of the language model pre-training, we experimented with how the model performs without the pre-training stage. This was done by skipping the pre-training stage and only train using the TACRED dataset to optimize the GPT architecture. We perform the experiments with and without the typed entity mentions strategy.

Fine-tuned GPT	F1
+ LM, + NE/GR	68.1
- LM, + NE/GR	64.0
+ LM, - NE/GR	63.2
- LM, - NE/GR	39.0

Table 5: Effect of pre-training on the fine-tuned GPT model meassured in micro-averaged F1. "+" marks that the given function is enabled, "-" marks it is disabled.

The results are shown in Table 5. We observe that the pre-training stage has quite a significant effect on performance since for the models with NE/GR included, the model with pre-training increases F1 score by 4.1 to 68.1%, which means that the pre-training does offer significant semantic understanding in addition to the EN/GR types. Looking at the variations without the NE/GR strategy, the addition of pre-training increases the F1 score by 24.2 to 63%. Which means that the pre-training itself is not enough to make the NE/GR strategy redundant, but does come close, which signifies that the language model pre-training does contain a significant level of semantic information also contained in the NE/GR strategy, useful in relation extraction.

8 Conclusion

In this project, we sought out to improve the task of relation extraction with the use of deep language representations of the type achieved by language models. We showed that the proposed model achieves a better F1 score than most other highly sophisticated neural systems on the dataset, without hand-designed explicit features. We, therefore, achieve two important milestones, a near state-of-the-art model in regards to F1 score and a system with a relatively simple data pipeline after the pre-training stage. This is due to the fine-tuning that does not require a complex pipeline of pre-processing, thus reducing the margin of error and simplifies the process of implementation in a production environment.

In the experiments, we saw that the importance of the pre-training stage is significant, by isolating its contribution in an ablation study. We also observed how the fine-tuning stage of the model is significantly data-efficient, meaning that little training data is required for the model to properly converge. We additionally, successfully

reproduced the results of Alt et al., who have also extensively experimented with the GPT on the TACRED dataset for relation extraction. While most results are on par with what they found, we further investigated the settings of the GPT for the relation extraction task, however, without any major concrete improvements. We ultimately had the same results of effective hyperparameters, although with a better results using a slightly higher learning rate, as noted in Table 3.

9 Future Work and Reflections

There is currently a very active field of research in using language models for relation extraction; which, the new additions of BERT models that outperform our fine-tuned GPT are testimony to. OpenAI recently published their newly trained GPT-2 model (Radford et al., 2018b) with a very similar architecture to the GPT-1 version. Initially, despite OpenAI being a front-runner in open-sourcing AI research, they did not intend to open-source the weights of the system, because of it being "dangerous" and a potential culprit for generating fake-news due to its generative ability to imitate human writing closely[4]. This begs the question concerning the rising power of these automated natural language processing systems and whether it is ethical to continue research on such systems when they can be used for such activities. While it is clearly possible to utilize such models for unethical activities, such as fake-news, it is also important to keep in mind the potential of ethical use-cases in these systems. Certainly, there are arguments on both sides of this case. Regardless of these arguments, monopolizing such powerful AI tools to only the largest tech companies would seem even more unethical and dangerous than developing the technology in the first place.

Recently, OpenAI contrary to their original opinion, decided to open-source the weights of the GPT-2. Unfortunately, work on this project was already started at that point, which is the primary reason this project is based on the previous version of the GPT. A GPT-2 based relation extraction system is, however, definitely something to be done in the future, as the pre-training stage of the GPT-2 is significantly broader and of a larger scale than that of GPT-1. As of yet, this is not something that has been pursued in academia in the context of relation extraction.

The motivation for this project, at its core, came from the necessity of relation extraction in the creation of knowledge graphs from a natural language corpus, specifically a multilingual corpus, containing both Danish and English text. At the time, the resources available to perform relation extraction in danish seemed rather sparse, partly due to lack of datasets and lack of natural language tools available, which is why English was chosen as the scope of this project. The findings of this project, however, begs the question of the possibility of an accurately functional model for minority languages such as danish. Since we have shown the GPT's capability of converging on fine-tuning with small amounts of data, although with proper pre-training, even a small supervised dataset in a minority language such as danish would likely be sufficient for the fine-tuning. However, the pre-training stage continues to be a challenge, but with the construction of a giant broad-coverage corpus such as the danish gigaword project (Strømberg-Derczynski et al., 2020), this task could end up being very realistic very soon. Ultimately, opening up the possibility for highly accurate language model based relation extraction.

10 Acknowledgements

A thanks to the Huggingface re-implementation of the original OpenAI GPT architecture. Which made the project possible[5]. This is in addition to the work of Alt et al. who also was a major inspiration in their work on the Huggingface source code and results on the GPT model.

[4] https://openai.com/blog/better-language-models/

[5] https://github.com/huggingface/transformers

References

Jay Alammar. 2019. The illustrated gpt-2 (visualizing transformer language models).

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving relation extraction by pre-trained language representations.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 81–88, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus).

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

D. Lukovnikov, A. Fischer, and J. Lehmann. 2020. Pretrained transformers for simple question answering over knowledge graphs.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2019. Bpe-dropout: Simple and effective subword regularization.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018b. Language models are unsupervised multitask learners.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea. Association for Computational Linguistics.

Min Song, Won Chul Kim, Dahee Lee, Go Eun Heo, and Keun Young Kang. 2015. Pkde4j: Entity and relation extraction for public knowledge discovery. *Journal of Biomedical Informatics*, 57:320 – 332.

Leon Strømberg-Derczynski, Rebekah Baglini, Morten H. Christiansen, Manuel R. Ciosici, Jacob Aarup Dalsgaard, Riccardo Fusaroli, Peter Juel Henrichsen, Rasmus Hvingelby, Andreas Kirkedal, Alex Speed Kjeldsen, Claus Ladefoged, Finn Årup Nielsen, Malte Lau Petersen, Jonathan Hvithamar Rystrøm, and Daniel Varab. 2020. The danish gigaword project.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540, Lisbon, Portugal. Association for Computational Linguistics.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 571–581, Vancouver, Canada. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3(null):1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.