Exploring pre-trained language models' capabilities of showing common-sense

by

Jerzy Bajer

A DISERTATION SUBMITTED IN PARTIAL FULLFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
COMPUTING
SCHOOL OF COMPUTING
EDINBURGH NAPIER UNIVERSITY
MAY 2022

Authorship Declaration

I, Jerzy Mariusz Bajer, confirm that this dissertation and the work presented in it are my own achievement.

Other authors' work that was used in this dissertation is attributed and the source is given. For all quoted and paraphrased work of other authors, the source is always given. All sources used in this dissertation have been acknowledged. With such exceptions this dissertation is entirely my own work.

I have read and understand the consequences and penalties of Academic Misconduct. While working on this dissertation I have followed the Napier University ethical guidelines.

Signed:	Jerzy Bajer
Date:	06.04.2022
Matriculation number:	40428545

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Jerzy Bajer

Abstract

This dissertation discusses the topic of pre-trained language models BERT and GPT-2 and their capabilities of showing common sense. This topic is very important because of the latest advancement in technology and AI that may allow people to communicate with computers using natural languages, for instance, English natural language.

This dissertation begins by introducing the topic. It introduces Natural Language Processing (NLP), common-sense, transformer and pre-trained language models BERT and GPT-2. It presents the aims and objectives and the structure of this project. It then presents the theory that helps to understand the BERT and GPT-2 models. It presents neural networks, word embedding, transformer, BERT, and GPT-2. It also compares BERT and GPT-2 and presents common-sense reasoning. Then this report presents the design and implementation of the models, that are used to test BERT's and GPT-2's capabilities of showing common sense. It briefly presents the steps required to create and use BERT and GPT-2 models.

Then this report presents the testing and the results of the BERT and GPT-2 models. The models' capabilities of showing common sense are tested with the usage of the prompt and predict technique. The prompt and predict techniques work the following way. The model gets the prompt, the question, and tries to answer it, tries to predict what text should go next as the continuation of the prompt. So, each model is asked questions (prompts), for instance: "What can be used instead of plain flour?" Each model answers the questions (predicts). The answers generated by each model are classified based on whether they match the correct answer which in this case is "oat flour, bread flour, cake flour or coconut flour " or whether they make sense. The report presents the analysis of the results and discusses the models and their capabilities.

This dissertation concludes that both BERT and GPT-2 models have some capabilities of showing common sense. Fine-tuned GPT-2 model seems to have the best capabilities of showing common sense from all the models presented in this dissertation. However, the capabilities of the models presented in this report are far from perfect. Therefore, further research and improvements are required to make BERT's and GPT-2's models' capabilities of showing common-sense closer to the human.

Contents

1. Introduction

Natural Language Processing is “the branch of artificial intelligence (AI) that deals with training a computer to understand, process, and generate language” (Bell & Olavsrud, 2021). The way the computers were programmed allow them to read, understand and process data that is represented in a structured way. For instance, computers can easily understand data in SQL database or JavaScript Object Notation (JSON) data. However, not all data can be represented in a structured way. The natural language is very complex. Ambiguity and lack of precise characteristics of the natural languages make it very difficult for computers to be able to read, understand and process natural language. For instance, the word “make” may have a different meaning based on the context in which it is used. If the word “make” is used in the sentence “Can you please make me a cup of coffee?” then the word is a verb and means doing something, in this case, a cup of coffee. However, using the same word “make” in the following sentence “What make is your car?” changes its meaning completely. In this context, the word "make" is used as a noun and means the brand of car. For human understanding such ambiguities are natural. Humans have common sense which “is an integral part of human cognition which allows us to make sound decisions, communicate effectively with others and interpret situations and utterances” (Clinciu, Gkatzia & Mahamood, 2021). However, computers do not have common sense. Therefore, for computers, it is very hard to see a difference in using the word "make" in two different contexts. The goal of NLP is to program computers “to understand, process, and generate language just like a person” (Bell & Olavsrud, 2020). The focus of NLP is to implement in computers human capabilities of reading, understanding, and processing natural languages. NLP tries to make computers be like a human with human capabilities. Implementing into computers common-sense, that humans have, would be a huge step forward. Computers' capabilities would be much closer to human capabilities.

Natural Language Processing is not something new, it was around for a very long time. NLP “began in the 1950s as the intersection of artificial intelligence and linguistics” (Nadkarni, Ohno-Machado & Chapman, 2021). However, recent advances caused a revolution in the development of natural language processing. First, computers are much more powerful than ever before, they have more random-access memory (RAM) and much more powerful microprocessors (CPU). This allows computers to gather and process huge amounts of data. Second, the algorithms and methods created to read, understand, and process natural language improved significantly. This “was enabled by recent advances in neural architectures, such as the Transformer” (Chernyavskiy, Ilvovsky, & Nakov, 2021) and the creation of pre-trained models such as BERT and GPT-2.

1.1 Problem statement

The creation of a transformer was a revolutionary achievement in natural language processing. Transformer allowed to creation of many great pre-trained language models. Two significant models, that were created based on the transformer, are BERT and GPT-2. BERT stand for Bidirectional Encoder Representations from Transformers, GPT-2 stands for Generative Pre-trained Transformer 2. Both BERT and GPT-2 have their advantages and disadvantages. So, the question that appears naturally is which of them is better? Answering this question is not easy as many criteria can be used to compare pre-trained language models. This dissertation focuses on the comparison of pre-trained language models such as BERT, GPT-2 and others and their capabilities of showing common sense.

1.2 Research questions

This project will try to answer the following questions:

- Which of the pre-trained language model BERT or GPT-2 has better capabilities of showing common-sense?
- Does fine-tuning the pre-train language model increase the model's capabilities of showing common sense?

1.3 Project's aim and objectives

This project aims to research, compare and test selected pre-trained language models. The first part of this project is the research in a form of a literature review. The second part is comparison and testing which includes coding.

The objectives of this project are:

- researching and understanding what the pre-trained language models are and how they work
- researching selected pre-trained language models (BERT and GPT-2), their strengths and weaknesses and comparing them.
- researching BERT and GPT-2 capabilities of showing common-sense.
- researching testing techniques of selected pre-trained language models (BERT and GPT-2)
- testing and comparing BERT and GPT-2 capabilities of showing common-sense.

1.4 The report's structure

This dissertation will be divided into the following chapters

1. **Introduction** – introduces the idea behind this project and outline the aim and objectives of this project.
2. **Literature Review** – research of selected pre-trained language models and their capabilities of showing common-sense.
3. **Design and Implementation** – explanation of how the common-sense capabilities will be compared using coding.
4. **Testing and Results** – presentation of the findings of the conducted tests.
5. **Analysis and Discussion** – evaluation and comparison of selected pre-trained language models and their capabilities of showing common-sense.
6. **Conclusion** – summary of the project's outcomes.

2 Literature Review

This chapter will start by presenting background information about neural networks and their role in natural language processing (Section 2.1). It will then explain words embeddings and their role in natural language processing (Section 2.2). The next section will focus on the transformer and different pre-trained language models that are based on the transformer (Section 2.3). This section will also present the comparison of the selected pre-trained language models, BERT, and GPT-2. Then there will be presented common-sense reasoning capabilities of selected pre-trained language models (Section 2.4).

2.1 Neural Networks

One of the biggest limitations of computers used to be the lack of computers' capabilities to learn. In the past computers could only do what they were explicitly told or more precisely programmed to do. In other words, computers could only solve simple problems following step-by-step instructions provided for them in a form of a computer program. This was good enough at the beginning. After some time though researchers noticed that some problems are too complex, and it is impossible to provide a set of simple instructions that would cover each possibility. For instance, natural language as was mentioned above is so complex that it is impossible to create a set of simple instructions to allow the computer to understand and process it. Researchers came up with the idea to teach computers to learn. In other words, researchers wanted to implement in computers human capability to learn.

Observation of the human brain and the way it works inspired the researchers. They came up with the idea of neural networks. The human brain "is a neural network" (Westland, 1998). The idea behind artificial neural networks was to imitate the human brain. The term "neural networks is used to describe a number of different models intended to imitate some of the functions of the human brain" (Westland, 1998). Neural networks are still far away

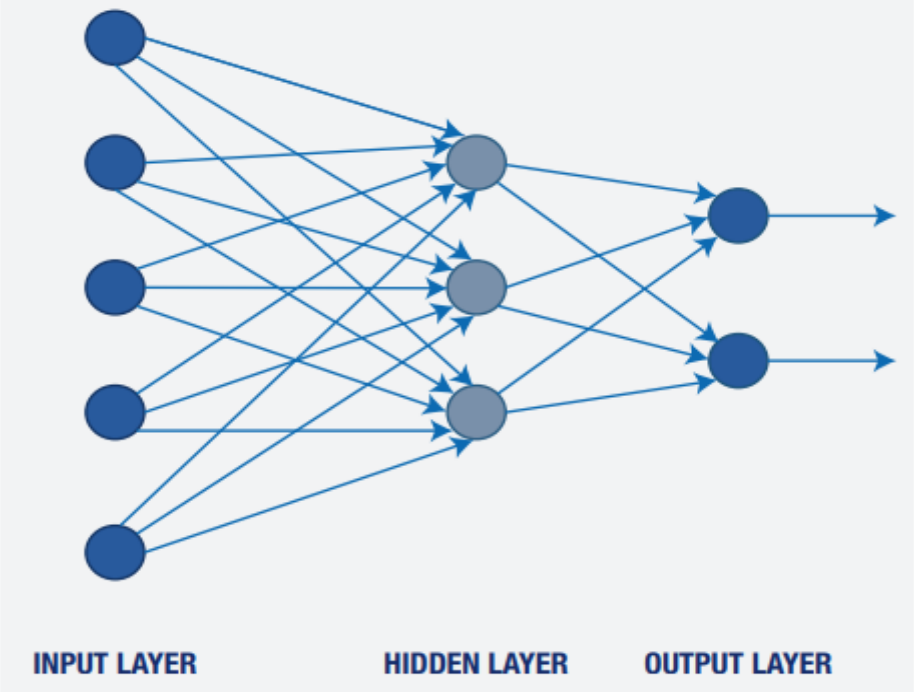
from the capabilities of the human brain, nevertheless, they changed the way natural language applications are created. Simply speaking neural networks are networks of connected neurons. Artificial neural networks consist of units called neurons. Somebody may ask then what is a neuron. A neuron in the human brain “receives information from other neurons (or from sensory cells such as those in the retina of the eye) using its dendrites and sends information to other neurons using its axons. The synapse is the point where the axon of one cell meets the dendrite of another” (Westland, 1998). In artificial neural networks, the neuron can be thought of as the smallest unit also called a node. Neuron in one layer receives information from neurons from the previous layer or user then it processes data and send it to the next layer.

The process of the creation of an artificial neural network can be divided into two major steps, training and testing. First, the artificial neural network must be trained with a set of training data. The training data set consists of input and the corresponding output. In “the learning phase the network is presented with known input-output” (Westland, 1998). During the training neural network model learns what output it should produce based on the given input. Once the training of the neural network model is done, the model can be tested. During the testing, the neural network model must produce output based on the given input. Then the output produced by the neural network model is compared with the expected output that should be produced. The closer the output produced by the model is to the expected output the better the model got trained and the higher the accuracy is. The accuracy tells how close the output produced by the neural network model is to the expected output.

It must be mentioned that the training data set should be like the testing data set otherwise the output produced by the model may be far away from the expected output, the model will have low accuracy. It seems to be obvious; it is like learning for an exam from

physics and then getting chemistry exam papers. Normal practice is to divide big data set into two parts training data set and testing data set. Common practice is the 80/20 division, which means that the data set is divided into 80% training data set and 20% testing data set.

The artificial neural network must consist of at least three layers: input layer, hidden layer, and output layer.



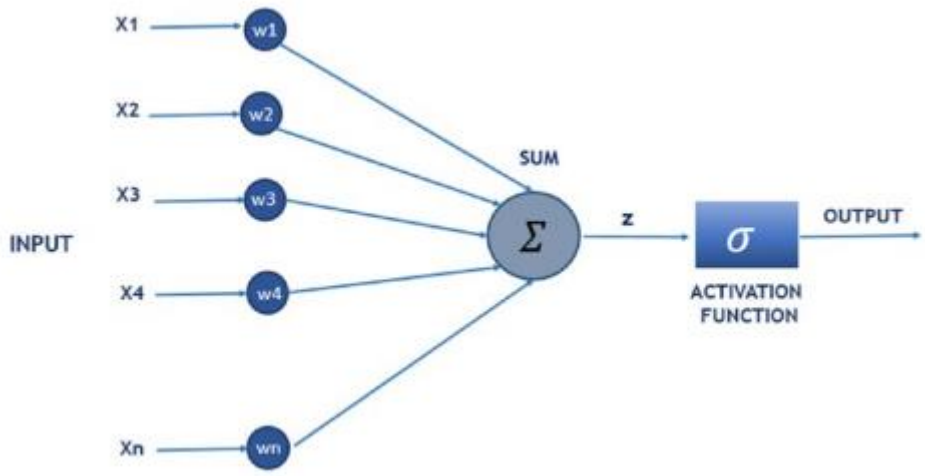
2.1.1. Neural network (Sabharwal and Agrawal, 2021)

The input layer is the first layer in every neural network model. The *input layer* "contains the input nodes interacts with the outside environment" (Alsultanny & Aqel, 2003). The input layer gets input data. Data can be pixels for images or numbers for a vector that represents a sentence for text. The number of nodes in the input layer must correspond to the size of the data. All features, which are represented by nodes, get assigned some weights. Weight specifies how important the feature is. Common practice is to assign a random value of weights to all nodes. The value of each weight will be adjusted during the training of the neural network model. Once weights are assigned to all nodes the output from the input layer is forwarded to the first hidden layer.

Each neural network model must have at least one hidden layer. There may be more hidden layers. The number of hidden layers depends on the task the neural network model must complete. There should be no more hidden layers than are required for the model to complete the task. Too many hidden layers will harm the performance of the model, it will take much longer to train the model. The number of nodes in the hidden layer is also specific to the task the neural network model must complete. Common practice is that the number of nodes in the hidden layer(s) is higher than the number of nodes in the input and output layer. Nodes in the hidden layer receive input from the previously hidden layer or the input layer then nodes process data and send output to the next hidden layer or the output layer. The activation function is implemented at the hidden layer.

The last layer is always the output layer. The purpose of the output layer is to gather all information from the last hidden layer and then to output the results. The number of nodes in the output layer depends on the task the neural network model must complete. For instance, for classification neural network model, the output layer represents “the number of nodes, which is equal to the number of classes” (Alsultanny & Aqel, 2003). Each output node represents a class.

Simple problems can be solved using traditional linear models. More complex problems require nonlinear neural network models to be solved. The activation function is used for neural network models that solve nonlinear problems. The activation function “used in neural networks calculates a weighted sum of the inputs and biases to determine whether and/or how much a neural unit should be activated” (Zhou, Huo & Kung, 2021). Speaking simply the purpose of the activation function is to calculate the sum of input values multiplied by corresponding weights and adding bias. Then based on the result, the activation function decides which feature should be passed to the next layer.



2.1.2. Activation function (Sabharwal and Agrawal, 2021)

There are two techniques used to train the neural network model: forward propagation and backward propagation.

Forward propagation means propagating forward. The activations in the hidden layers depend on the input that was passed. Different inputs will activate different neurons in the hidden layer. Then different activated neurons in the hidden layer will affect neurons in the next hidden layer. This will affect the neurons in the output layer and the output generated by neurons in the output layer. So, the final output is produced by propagating the input from the input layer through the hidden layers to the output layer.

Backward propagation means propagating backwards. The actual output values are compared with the output values predicted by the neural network model. This allows the calculation of the difference between the predicted output values and the actual output values. This difference can be considered as an error. This error then is passed backwards from neurons in the output layer to the neurons in hidden layers and allows to update all weights and bias. The purpose of updating the weights and bias is to train the model so that the predicted outcome values match the actual outcome values.

Forward and backward propagation is often used together to train the model. The training starts with forward propagation. Based on the input the neural network tries to predict the output. Random weights and biases are assigned to the input at the input layer. Then data is passed to the hidden layer. Different input activates different neurons in hidden layers which affect the predicted values produced by the neurons in the output layer. Once the model produce predicted outcome values the forward propagation ends. The predicted outcome values are compared with actual outcome values. If the predicted outcome value and the actual outcome value are different then the backward propagation process starts. Comparison of the expected and predicted outcome values allows calculating the error. Then the error is passed backwards, and weights and bias are updated appropriately to improve the model capability to predict the outcome values that will match the actual outcome values. Repeating this process allows training the model so that the model predicts outcome values which will be as close as possible to the actual outcome values.

There are different types of neural networks.

The most basic type of neural network is feed-forward neural network (FFNs). As the name suggests it is one direction (forward only) neural network.

Convolutional Neural Network (CNN) is a specific type of neural network that is very useful for image classification. CNN “uses convolutional layers to explore spatial/temporal adjacency to construct new feature representations “(Han & Zhu, 2019).

For natural language processing, the most useful type of neural network is Recurrent Neural Networks (RNN) and their variants. This type of network can capture contextual dependencies of data. RNNs and their variants “have been used in many contexts where the temporal dependency in the data is an important implicit feature in the model design” (Bianchi et al., 2017). One of the most popular variants of the RNN neural network model is Long Short-Term Memory (LSTM) variant.

The encoder-decoder infrastructure consists of two RNNs, the encoder, and the decoder. The encoder is long short-term memory (LSTM), its goal is to get the input and encode it into an internal representation. The decoder on the other hand takes the encoded sequence produced by the encoder and based on it generates the output. Based on the produced by encoder internal representation decoder tries to make the prediction.

Bidirectional encoder-decoder architecture consists of bidirectional LSTM (encoders and decoders). In bidirectional encoder-decoder architecture, the “last hidden state of the backward encoder initializes the forward decoder, whereas the backward decoder is initialized with the last hidden state of the forward encoder” (Sabharwal and Agrawal, 2021). The encoder which is LSTM works both forward and backward. The decoder consists of two LSTMs one works forward, and one works backwards.

Understanding the neural networks and the way neural networks work is vital because both pre-trained language models such as BERT and GPT-2, which are the subject of this project, are neural networks models that are based on the transformer. Transformer, BERT, GPT-2, and other pre-trained language models will be discussed in Section 2.3 as was mentioned above. So, both pre-trained language models BERT and GPT-2 are based on Transformer. The transformer is the neural network model that is based on bidirectional encoder and decoder infrastructure. Encoder and decoder infrastructure as it was mentioned is based on Long Short-Term Memory (LSTM). Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Networks (RNN) which is one of the types of neural networks.

2.2 Word Embedding

Natural language is very complex. Natural language processing focuses on implementing in the computers human's capability to read, understand and process natural language in a form of textual data. Implementing this capability depends on the way the textual data is represented. Neural network models that are used for natural language processing require textual data to be represented in a numerical form. Converting textual data in a numerical form is a challenging task. There are different techniques used to convert textual data into numerical form.

2.2.1 One-hot representation

One-hot representation is one of the simplest techniques that allows representing textual data in a numerical form. This technique uses "a large size of the vector and assumes that the features that make up the vector are independent of each other" (Song et al., 2018). The size of the vector is equal to the size of the vocabulary. The one-hot representation uses two binary values 0 and 1 to determine the absence or presence of a word in the vocabulary. A simple example will help to understand how this technique works. For the sentence "I like football I dislike volleyball" the vocabulary will have five words [I, like, football, dislike, volleyball]. Each word can appear in vocabulary only once.

Now for each word will be created a vector showing presence of a word in the vocabulary.

[1, 0, 0, 0, 0] – word "I" is present in vocabulary.

[0, 1, 0, 0, 0] – word "like" is present in vocabulary.

[0, 0, 1, 0, 0] – word "football" is present in vocabulary.

[0, 0, 0, 1, 0] – word "dislike" is present in vocabulary.

[0, 0, 0, 0, 1] – word "volleyball" is present in vocabulary.

As it can be seen one-hot representation is an easy technique to use. The simplicity of one-hot representation defines the drawbacks of this technique. One-hot representation only pays attention to whether the word is present or not. This is the biggest drawback. One-hot representation does not show the occurrence of a word. Certain words may be present more often in a sentence than others.

2.2.2 Count vector

Count vector is a bit more complicated technique that works like one-hot representation except that it shows the occurrence of each word not just the presence or absence. For the same sentence “I like football I dislike volleyball.” the count vector will look like this.

[2, 0, 0, 0, 0] – word “I” is present twice.

[0, 1, 0, 0, 0] – word “like” is present once.

[0, 0, 1, 0, 0] – word “football” is present once.

[0, 0, 0, 1, 0] – word “dislike” is present once.

[0, 0, 0, 0, 1] – word “volleyball” is present once.

Count vector is the same as one-hot representation is a very easy technique to use, and it also has its limitations. The one-hot representation shows only the presence of the word. Count vector that is a bit more complex technique goes a bit further and shows the occurrence of a word. But neither a one-hot representation nor the count vector shows the importance of the word in a sentence.

2.2.3 TF-IDF

Certain words may occur in a sentence many times, but their importance may be very low or none. Other words on the other hand may appear not so often, but their importance may be the key to understanding the sentence. TF-IDF is a technique that shows the importance of a word. TF-IDF stands for term frequency-inverse document frequency. TF–

IDF “representation is based on the bag-of-words philosophy, which involves the assumption that a document is simply a collection of words, and thus, the document can be vectorized by computing the relative importance of each word, i.e., by considering the word’s frequency in the document and its popularity in the corpus” (Kim et al., 2019).

2.2.3 Word embedding

The one-hot representation shows only if a word is present or not, the count vector shows the occurrence of the word. TF-IDF which is much more complicated than the presented one-hot representation and count vector also shows the importance of the word. However, none of the presented so far techniques capture the contextual meaning of the word. As mentioned earlier, the context in which a word is used may change the meaning of the word. For instance, word make based on context may mean doing something or may mean a brand. Word embedding, also known as a distributed semantic model, is a technique that tries to capture the context in which a particular word was used.

Word embedding is a vector representation of words that is used for modern machine learning for natural language processing (Turney & Pantel, 2010). Words with similar meanings should have similar vector representations. So, words with similar meanings should be placed close to each other. For instance, words that describe animals such as dog, cat, or pig should be placed close to each other, whereas words with completely different meanings should be placed far away from each other. The goal is to place words with similar context together. Word embeddings have different methods.

Word2vec is a two-layered shallow word embedding method that has input only one hidden layer between the input layer and output layer. Word2vec method represents “words in vector space based on unsupervised prediction and aims to minimize the distance value of words that are semantically and syntactically close” (Dev, Hassan & Phillips, 2021). Word2vec puts the words with a similar semantic meaning close to each other and words

with different semantic meaning far away from each other. This is achieved by the creation of a semantic relationship between the words based on the context in which each of the words was used. For instance, for the sentence “What is the time?”, the context of the sentence is that someone wants to know the time. Capturing the context of the sentence and the words is the key of the Word2vec method. Word2vec can be implemented in two ways: using a continuous bag of words (CBOW) or skip-gram.

Continuous bag of words (CBOW) is an efficient “shallow neural network algorithm. It is developed to generate vector representations of a language vocabulary so that the information of the words is encoded in the vector space structure” (Mikolov et al., 2013). CBOW tries to predict the word based on the context of the words that surround this word. CBOW uses the one-hot representation for its input layer.

Skip gram is a technique that tries to “capture syntactic and semantic information about words without any explicit supervision in this respect” (Kocmi & Bojar, 2018). The prediction of the word is based on the most relevant words that surround the word. This technique is a bit more difficult to be implemented than a continuous bag of words.

Global vector (GloVe) is an unsupervised learning method (Pennington, Socher & Manning, 2014), that is used for word embedding. This method creates word vectors by combining global and local contexts of the word. Local context focuses on the words that surround the word. Global context focuses on the occurrence of the word. Word2vec pays attention only to words that surround the word. GloVe on the other hand not only takes into consideration the words that surround the word but also the global occurrence of the word.

Word embedding allows to capture the relationships between words in a sentence, however, word embedding lacks the capability to capture the relationships between the sentences, especially when both sentences have the same word representations. The method that allows capturing the relationships between the sentences is sentence

embedding. Sentence embedding “transforms sentences into low-dimensional vector values reflecting their meanings” (Jang & Kang, 2020). A simple example will help to understand the difference between two sentences with the same word representation and different meanings. For instance, there are two sentences: “Today is Monday, not Tuesday” and “Today is Tuesday, not Monday”. Both sentences have the same words, but completely different meanings. The order in which the words were used affects the meaning of each sentence. Word embedding will create two pretty much the same representations one for each sentence. As a result, word embedding will not be able to notice the different meanings of the sentences. Sentence embedding is useful just for such situations as it allows to capture that despite both sentences having the same word representations, they have a different meaning.

Embeddings from Language Models (ELMo) is a very efficient method that is used for word embedding. ELMo has “recently been shown to model word semantics with greater efficacy through contextualized learning on large-scale language corpora, resulting in significant improvement in state of the art across many natural language tasks” (Tseng, Georgiou & Narayanan, 2019). ELMo uses a neural network learning model to create vectors for words. ELMo consists of two bidirectional LSTM neural network models that are used to create word representation. One LSTM allows to ELMo to understand the previous word in a sentence and another LSTM allows to understand the next word in the sentence. This way ELMo can capture the context in which each word was used in the sentence. ELMo has the capability to learn language patterns. Thanks to this capability ELMo can guess which word can be put in the missing space in a sentence. For instance, there is a sentence “It is morning, the sun ...”. As it can be seen one word is missing. ELMo can search the prior words history and based on language pattern ELMo can decide to put the word “rises”. This way the sentence will be “It is morning, the sun rises”, which makes sense.

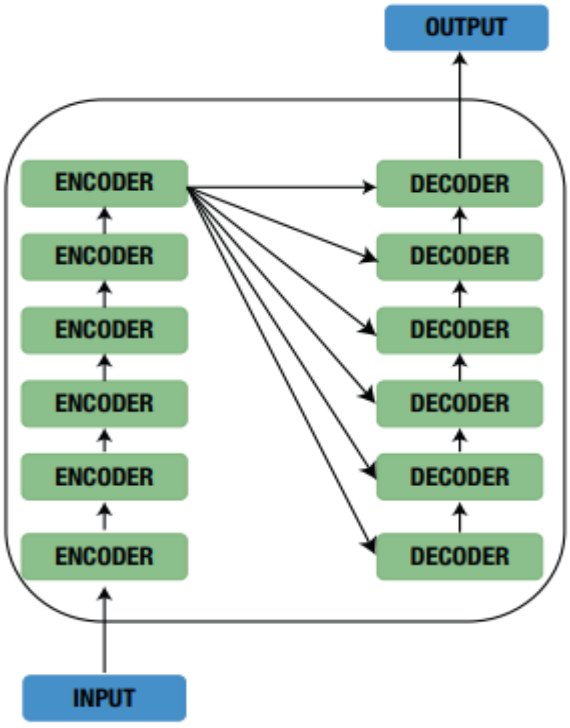
Techniques that were presented in this section are not all techniques that are used to convert textual data into numerical forms. They are the most common techniques used. The next section will start by presenting the transformer. When presenting transformer there will be presented Universal Sentence Encoder, the technique that is used by the encoder in the transformer.

2.3 Neural Network Models

The invention of the pre-trained language model BERT in 2018 was a revolutionary step for NLP (Devlin et al., 2019). As it was mentioned, both BERT and GPT-2 are based on Transformer. The invention of the Transformer enabled this revolutionary step (Vaswani *et al.*, 2017). This section will start with the presentation of the transformer. Then there will be presented BERT and GPT-2 and other pre-trained language models based on BERT.

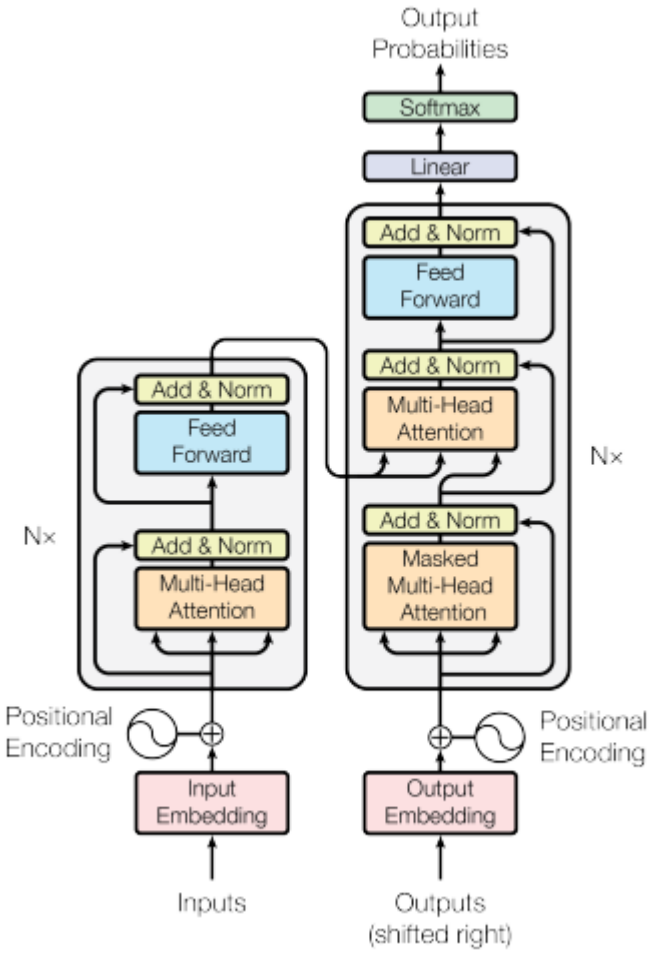
2.3.1 Transformer

Transformer “is an autoregressive encoder–decoder model using a combination of multiheaded attention layers and positional feed forward to solve sequence-to-sequence (seq2seq) tasks” (Chen *et al.*, 2020).



2.3.1.1 Transformer structure. (Sabharwal and Agrawal, 2021)

The transformer consists of an even number of stacked encoders and decoders. The number of encoders and decoders can vary if the number of encoders is equal to the number of decoders. Both encoders and decoders have pretty much the same structure, they are stacked in a structure of self-attention and fully connected layers.



2.3.1.2 Transformer architecture (Vaswani et al., 2017)

The key to understanding the transformer is understanding attention. The textual data must be converted into numerical form before it can be sent to the neural network model. Transformer uses a word embedding technique that is called the universal sentence encoder technique. Universal sentence encoder (USE) “is an embedding technique with the notion of learning to map fixed-length sentences” (Kumar et al., 2021). This technique allows converting any sentence to a fixed-length numerical vector. Universal sentence encoder “encodes any a given sentence to exactly 512 dimension embedding, irrespective of the sentence’s length” (Kumar et al., 2021). Conversion of sentences of different sizes into vectors of the same length may result in loss of contextual information. For instance, there are two sentences one sentence is very long and has 30 words, another on the other hand is short and has only 5 words. If both sentences will be converted to the same fixed-length

vectors, then some contextual data can be lost. This is where attention plays an important role. The self-attention mechanism is part of the attention sublayer, and it plays a very important role in transformer as it allows to solve data loss problems.

The self-attention mechanism allows to “extract a wide range of relevance information in the text” (Li et al., 2021). The self-attention mechanism allows each word in the input to associate itself with other words in the input. The self-attention mechanism takes input data and ads attention weights and produces an output vector with contextual data showing contextual relationships between words. Attention sublayer is where the power of transformer lays.

As mentioned, a transformer consists of an even number of encoders and decoders. Each encoder has the same layer structure and consists of two sublayers. The first sublayer of the encoder is a multi-head attention layer that contains a self-attention mechanism, the second sublayer of the encoder is a fully connected feed-forward neural network (FNN) model. Both sublayers are connected by residual connections together with layer normalization. The textual data is embedded into numerical form and goes through a self-attention mechanism where attention weights are added and go to the feed-forward neural network (FNN). The goal of the encoder is to encode the input data into a continuous representation with added attention information. This output data produced by the encoder helps the decoder to focus on certain words during the decoding process. Each decoder has the same structure and consists of three sublayers. The first sublayer of each decoder is the masked multi-head attention layer which contains a self-attention mechanism. The mask prevents words from reaching the next word in the input. Only previous words can be accessed. The second sublayer is an additional multi-head attention layer. The purpose of the additional multi-head attention layer is “to extract interaction information between the encoder and the decoder“(Chen *et al.*, 2020). The second multi-head attention sublayer

takes the output from the first multi-head attention sublayer and the output from the encoder. The decoder matches the data from the encoder with data from the first multi-head attention sublayer and decides which data from the encoder is relevant and should be focused on. The output goes then through a feed-forward neural network (FNN) model where is further processed. Then the output goes to the linear layer which works as a classifier and allows the classification of a word based on the number of classes. The softmax allows producing probability scores between 0 and 1 for each class. The index with the higher score is the predicted word.

This is how the transformer works. The next subsections will focus on presenting BERT and GPT-2, two pre-trained language models based on the transformer. Then there will be presented other pre-trained language models based on BERT.

2.3.2 BERT

Bidirectional Encoder Representation from Transformer (BERT) is based on the transformer. As it was mentioned in the previous subsection, a transformer consists of an even number of stacked encoders and decoders. BERT needs only encoders. BERT consist of a stacked number of encoders, that allow to read sentence bidirectionally, from left to right and from right to left, and embed each word into a vector that contains word meaning based on the context in which the word is used. For instance, the vector for the word “like” in the sentence “I like football” will be different from the vector for the word “like” in the sentence “she looks like her mother”.

BERT is a neural network model, so before textual data can be fed into an encoder it must be converted into vectors of numerical values. BERT uses a combination of position embedding, segment embedding, and token embedding. Position embeddings allow us to learn the order in the embeddings. Learning the position of words in a sentence is important because word “ordering often determines the meaning of a sentence” (Wang & Chen, 2020). Segment embedding allows learning unique embedding that allows distinguishing between sentences. Finally, token embedding allows to “capture linguistic characteristics expressed in the context of a token” (Tu, Gimpel & Livescu, 2017). All three combined techniques allow the preparation of textual data for the BERT model.

BERT model allows accomplishing a variety number of NLP tasks such as language translation. For instance, text can be translated from French into English or vice versa. BERT model also allows creating question answering systems, sentiment analysis, text summarization, and many more. Accomplishing any NLP task with BERT requires two steps. First, the BERT model must be pre-trained. The pre-training phase focuses on teaching the BERT model to understand the natural language and the contextual meaning of words as words may have different meanings based on the context in which they are used. Once the model gets pre-trained it must be fine-tuned. This phase focuses on teaching the BERT model how to accomplish a specific NLP task, for instance how to answer questions. A simple example will help to understand it. For instance, the NLP task is to create questions answering system in the English language. First, the BERT model must be pre-trained, so it understands the English language. Once pretraining is finished the model must be fine-tuned in other words must be taught how to answer questions.

The goal of pre-training the model is to teach the model to understand the language and the context. BERT model learns the language by training simultaneously on two tasks.

The first task used to pre-train BERT model is the Masked Language Modelling (MLM) task that allows the BERT model to be trained to “predict a random sample of input tokens that have been replaced by a [MASK] placeholder in a multi-class setting over the entire vocabulary” (Yamaguchi et al., 2021). A simple example will help to understand it a bit better. For instance, there is a sentence “man is sitting on the bench” and the random word “sitting” is masked. So, the sentence looks like this “man is [MASK] on the bench”. Now, based on all the words being available and the context in which each word was used the BERT model must predict which word should be put in place [MASK]. The words are masked randomly. The number of words that are masked in a sentence must be small, otherwise, the model will not have enough contextual information to do the prediction. For instance, if every second word in a sentence will be masked then the model will struggle to predict which word can be used. Masked Language Modelling is a very easy and efficient way to train the model, it also allows the BERT model to understand and learn the contextual relationships between words in a sentence.

The second task that is used to pre-train the BERT model is the next sentence prediction task. This task focuses on training the BERT model to predict the next sentence in the text. Next sentence prediction task “inputs two sentences A and B into BERT at the same time to predict whether sentence B comes after sentence A in the same document” (Sun et al., 2021). A simple example will help to understand how the next sentence prediction task works. For instance, there is some text. There are taken two sentences where the second sentence comes after the first sentence and those sentences are put as the first pair. There are also taken two random sentences where the second sentence does not come after the first sentence and those two, not related sentences are also put as the second pair.

I pair sentences:

A: “Monika woke up in the morning.”

B: “She looked at the clock to check what the time is”

II pair of sentences

A: “Tom likes cars.”

B: “Today is beautiful weather.”

Then each pair of sentences is sent to the BERT model to predict if sentence B comes after sentence A in the text. As it can be seen in the example in the first pair the sentence B comes after sentence A, however in the second pair there are two different not related to each other sentences. The next sentence prediction task is a binary type of task where the model must predict 1 or 0. 1 means that the second sentence comes after the first sentence and 0 means that the second sentence does not come after the first one. The next sentence prediction task is an easy and efficient way to train the model and allows the BERT model to understand and learn about the relationships between sentences in the text.

Using both the masked language modelling (MLM) task and the next sentence prediction (NSP) task simultaneously allows BERT to get a good understanding of language.

Once the BERT model gets pre-trained it must be fine-tuned for specific NLP tasks. To use BERT for a specific task a small layer must be added to the output. For instance, to train the BERT model for the classification task, there must be added a small layer that will take the BERT model’s output and classify it.

There are two BERT models, BERT base and BERT large. BERT base model has 12 layers (transformer blocks), 768 dimensions, 12 multi-head attention layers, and 110 M. total parameters. BERT large has 24 layers (transformer blocks), 1024 dimensions, 16 multi-head attention layers.

BERTBASE.
Number of Layers of the transformer blocks = 12.
Output Dimensions = 768.
Number of Multi headed Attention = 12.
Total Parameters = 110 M.
BERTLARGE.
Number of Layers of the transformer blocks = 24.
Output Dimensions = 1024.
Number of Multi headed Attention = 16.
Total Parameters =

2.3.2.1 BERT base and BERT large models (Geetha & Karthika Renuka, 2021)

2.3.3 GPT-2

GPT-2 is a pre-trained language model that was created by OpenAI based on the GPT. The GPT-2 is a successor of GPT. GPT-2 can be considered as “a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages” (Radford, 2021). The main goal behind the creation of the GPT-2 was to allow the model to predict “a next word token given previous tokens and context” (Kieuvongngam, Tan & Niu, 2020). For instance, based on the given text: “John is six feet”, the GPT-2 model can predict that the next word should be “tall”. The sentence will be “John is six feet tall”. As it was mentioned GPT-2 is a large transformer, the large dataset was used to train GPT-2. The usage of a huge dataset allows GPT-2 to accomplish many other NLP tasks, for instance, GPT-2 can generate text based on a given input. For instance, based on the given text: “Julie left home at 6 am”, the GPT-2 model can create the continuation which may look something like this “to get the bus to work. She did not want to be late as it was a very important day at work and being late could result in serious consequences.” Of course, this is only a simple example showing the GPT-2 capabilities. One of the characteristics of the GPT-2 model is the fact that it does not need a domain-specific dataset because of the size of the dataset used to train GPT-2. Other NLP tasks where GPT-2 model finds use are questions answers systems, summarization tasks, and translation tasks. GPT-2 does not

require a task-specific dataset. All it requires is sufficient data and computation power. However, GPT-2 model can be further trained “with an additional custom dataset using a method called transfer learning to produce more relevant text” (Whitfield, 2021).

A publication called “Generating Wikipedia by Summarizing Long Sequences” presents the model based on transformer-decoders only, in which the encoders were dropped. The authors of this publication introduced “a simple but effective modification to T-ED for long sequences that drops the encoder module (almost reducing model parameters by half for a given hyper-parameter set), combines the input and output sequences into a single ”sentence” and is trained as a standard language model” (Liu et al., 2018).

GPT-2 model is such a model that requires decoders only; therefore, it is sometimes called a decoder-only transformer, the encoders are dropped. The GPT-2 model consists of the number of decoders stacked one on top of another. The number of decoders in the GPT-2 model depends on the variant. There are four GPT-2 model variants, small, medium, large, and extra-large. Each variant has a different number of decoders, number of parameters, and dimensions. Table 2.3.2.1 presents all GPT-2 models with the number of parameters.

Model	**Parameters**
Small	124M
Medium	355M
Large	774M
Extra Large	1558M

2.3.3.1 GPT-2 models, sources: (Ziegler, 2020), (Clark, 2021), (Solaiman, 2021)

The models were released in stages, starting from small and ending with extra-large. The reason behind the staged release was “to give people time to assess the properties of these models, discuss their societal implications, and evaluate the impacts of release after each stage” (Radford, 2021).

As it was mentioned the encoder has two layers: a multi-head attention layer that contains a self-attention mechanism and a feed-forward neural network layer. The decoder on the other hand has three layers. The first layer is the masked multi-head attention layer that contains the masked self-attention mechanism then is the multi-head attention layer that contains the encoder-decoder self-attention mechanism and finally, there is the feed neural network layer. The decoder used in GPT-2 model does not need a multi-head attention layer that contains an encoder-decoder self-attention mechanism because there is no encoder used in GPT-2.

GPT-2 is autoregressive. This means that the GPT-2 model generates only one token (one word) at a time. A simple example will make it easier to understand. For instance, there is a sentence “Mary woke”. This sentence is fed as input to the GPT-2 model. Same as with other neural network models, the GPT-2 requires that each word, each token be represented in a numerical form. The model takes the input tokens then based on the contextual meaning of each token the model predicts the next token (next word) as “up”. Then the model outputs the “up” token. The next step is adding the “up” token to the end of the sentence. The new sentence is “Mary woke up”, which is then fed again as new input. The model predicts the next token, however, this time the model takes into consideration the three tokens, “Mary”, “woke” and “up”. The process repeats itself over and over, each time the model takes the input, predicts the next token based on the input, outputs the token which is added to the input and the process starts again. The drawback of autoregression is that GPT-2 model can incorporate context on one side only. It can only access previous tokens.

As it was mentioned the decoder has a masked self-attention layer. The masked self-attention mechanism that is used in the decoder (GPT-2) works slightly differently than the self-attention mechanism used in the encoder (BERT). The masked self-attention mechanism used allows picking only those tokens that are to the left from the token position

for which the masked self-attention calculation is made. The normal self-attention mechanism allows picking tokens on both sides of the token position for which the self-attention calculation is made. However, tokens to the right are replaced with [MASK]. A simple example will make it easier to understand. For instance, there is a sentence “Today is Friday and tomorrow is Saturday” The model wants to make the self-attention calculation for the token “and”.

For the multi-head attention layer used in the encoder (BERT) it will look like this:

<s> Today is Friday and [MASK] [MASK] [MASK]

For masked multi-head attention layer in the decoder (GPT-2) it will look like this:

< s > Today is Friday and

As it can be noticed normal self-attention replaces tokens to the right with [MASK] while masked self-attention mechanism blocks access to all tokens to the right. Therefore, BERT is called bi-directional (both ways), it can go to the left and to the right to access each surrounding token (word) on both sides to capture the contextual meaning of the current token (word). GPT-2 is uni-directional (one way), can only go and access previous tokens to capture the contextual meaning of the current token (word).

2.3.4 Other pre-trained language models

The invention of BERT was a revolutionary step for NLP. As it was mentioned BERT was built based on the transformer. The transformer was a foundation for BERT, BERT is the foundation for pre-trained language models such as ALBERT and RoBERTa. Both of those models are variants of BERT.

ALBERT stands for “A Lite BERT”. ALBERT can be considered as one of the natural successors of BERT. ALBERT was built based on BERT and the same as BERT ALBERT

consists of several transformer encoders stacked one on the top of another. ALBERT requires much less computation power compared to BERT. It is a lite smaller version of BERT. ALBERT is “a variant of BERT with cross-layer parameters sharing and factorized embedding parameterization” (Chiang, Huang & Lee, 2020). Factorized embedding parameterization and cross-layer parameters sharing are two techniques that allow parameter reduction. Reducing the number of parameters allows ALBERT to scale up easier. ALBERT uses the parameters more efficiently than the BERT. First, the factorized embedding parameterization technique allows to decompose “the large vocabulary embedding matrix into two small matrices” (Lan et al., 2019). This way the size of the hidden layers can be separated from the size of the vocabulary. This results in reduced number of parameters which has positive impact on scalability. It is much easier to scale up the model. The growth of the hidden layers does not cause significant growth the vocabulary because they are separated. Second, the cross-layer parameter sharing technique “prevents the parameter from growing with the depth of the network” (Lan et al., 2019). It also makes the transition from layer-to-layer smoother. Using both techniques allow to significantly reduce the number of parameters without significant drop of the performance. ALBERT model like “BERT-large has 18x fewer parameters and can be trained about 1.7x faster” (Lan et al., 2019). ALBERT uses MLM (masked language modelling) for training the model same as BERT but does not use NSP (next sentence prediction). Instead of NSP, ALBERT uses sentence order prediction (SOP), which is new training method developed for ALBERT.

RoBERTa stands for “Robustly optimized BERT pre-training approach” and is an optimized way to train the NLP model. Both RoBERTa and ALBERT are “variants of BERT, and they have shown better performances than BERT in various tasks “(Liao et al., 2020). RoBERTa the same as BERT consists of the number of stacked encoders one on top of another. However, RoBERTa introduced certain improvements to BERT architecture.

RoBERTa “is optimized on the basis of BERT model by changing static masks to dynamic masks, improving the text encoding, and training with larger batches” (Liao et al., 2020). BERT uses static masking; words in a sentence are masked during the pre-processing phase. Feeding the same sentence multiple times results in that the same words are always masked. RoBERTa uses dynamic masking, words are masked when the sentence is fed. Feeding the same sentence multiple times results in that the different words being masked. By masking different words in the same sentence each time, the sentence is fed, a dynamic mask duplicates the training data. Using a dynamic mask improves the performance and gives better results than using the static mask. RoBERTa uses much more data to pre-train the model than BERT. RoBERTa trains the model on much longer sequences than the BERT. Using a bigger batch size to train the model brings better results.

Presented models ALBERT and RoBERTa are not the only models that are variants of BERT or that are based on BERT. There are other models such as StructBERT or DistilBERT. Presenting all models is beyond this work.

2.3.5 Comparison

The comparison will focus on comparing BERT with GPT-2. Models such as ALBERT and RoBERTa, which are variants of BERT, will be omitted. Those models are BERT models with some modifications and improvements. Those modifications and improvements were presented in the previous subsection 2.3.4 Other pre-trained language models. There is no reason to repeat all those information.

Both BERT and GPT-2 are based on the transformer, however, BERT consists of the number of encoders stacked one on top of another, and GPT-2 consists of the number of decoders stacked one on the top of another. As was mentioned above the structure of the

encoder and decoder is different. The encoder consists of two layers (multi-head attention layer with self-attention mechanism and a feed-forward neural network layer), the decoder used in transformer consists of three layers (the masked multi-head attention layer that contains the masked self-attention mechanism, the multi-head attention layer that contains the encoder-decoder self-attention mechanism and the feed neural network layer). However, the GPT-2 requires decoders with only two layers that are the masked multi-head attention layer that contains the masked self-attention mechanism and the feed neural network layer. The multi-head attention layer that contains the encoder-decoder self-attention mechanism is not required because the GPT-2 does not use encoders.

The GPT-2 is autoregressive, BERT is not autoregressive. Autoregression means that GPT-2 can output only one token at a time as it was explained during presenting GPT-2. BERT does not have such limitations. The consequence of autoregression allows GPT-2 to get the context of the word only based on the tokens to the left from the current token. BERT which is not autoregressive on the other hand can get the context in which the word (token) was used based on the words (tokens) on both sides of the current token (to the right and the left from the current token). BERT is bidirectional, it can go to the left and to the right, GPT-2 is uni-directional it can only go to the left.

BERT uses a self-attention mechanism, GPT-2 uses a masked self-attention mechanism that works slightly differently. The self-attention mechanism used in encoders in BERT prevents access to the future tokens by masking them, replacing them with [MASK]. So those tokens can be accessed, but those future tokens were replaced with [MASK]. The masked self-attention mechanism used in decoders in GPT-2 prevents access to the future tokens not by changing the future tokens to [MASK] but by blocking the information from future tokens.

Both BERT and GPT-2 can be used for a wide variety of the NLP tasks

Table 2.3.5.1 presents a summary comparison of BERT and GPT-2

BERT	**GPT-2**
Based on transformer	Based on transformer
Encoders stacked one on top of another	Decoders stacked one on top of another
Not autoregressive	Autoregressive
Bi-directional	Uni-directional
Multi head attention layer	Masked multi-head attention layer
Can be used for different NLP tasks.	Can be used for different NLP tasks.

2.3.5.1 Summary comparison of BERT and GPT-2

2.4 Commonsense reasoning

As it was mentioned common sense is part of human cognition and allows humans to make sound decisions, communicate effectively with others (Clinciu, Gkatzia & Mahamood, 2021). Common sense allows humans to get the context of utterance, capture irony and sarcasm. Implementing the common-sense capabilities into computers requires computers to learn and understand the language first. Implementing the common-sense capabilities into computers would be a huge step forward in AI. Computers' capabilities would get much closer to human capabilities, computers would become intelligent.

Some basic forms of common sense were implemented into computers with the usage of external knowledge bases. However, as it was mentioned the language is very complex. And providing the external knowledge bases is not the best solution. A much better solution would be teaching computers to learn and understand the natural language itself through training and gaining common-sense capabilities during the training.

The focus of this project as was mentioned above is researching and comparing the selected pre-trained language models and their capabilities of showing common sense. The pre-trained language models such as BERT and GPT-2 managed to teach the computer to learn and understand the natural language. The questions are, do the BERT and GPT-2 have common-sense capabilities? The next subsection will try to answer those questions based on available literature. The practical part of this project will try to show what it looks like in reality.

2.4.1 Pre-trained models and their common-sense reasoning abilities

Both BERT and GPT-2 have some basic common-sense capabilities. They both can for instance capture the contextual meaning of the word in a sentence. They both can predict the next word in a sentence. However, for instance, BERT's common-sense capabilities

“may not work for numerical commonsense knowledge (e.g., a bird usually has two legs)” (Lin et al., 2020).

There was experiment, a numerical dataset was created to check the numerical common-sense capabilities of different pre-trained language models. Table 2.4.1.1 presents some examples of this dataset.

Category	Example
Objects (35.2%)	A bicycle has *two* tires.
Biology (13.5%)	Ants have *six* legs.
Geometry (11.7%)	A cube has *six* faces.
Unit (6.3%)	There are *seven* days in a week.
Math (7.3%)	I will be *ten* next year, as I am *nine* now.
Physics (5.7%)	Water will freeze at *zero* degrees centigrade.
Geography (2.9%)	The world contains *seven* continents.
Misc. (17.5%)	There are *no* princes in the United States.

2.4.1.1 Numeric dataset examples (Lin et al., 2020)

For this experiment, there were used different pre-trained language models, such as GPT-2, BERT. There was also used RoBERTa a variant of BERT. Finally, this experiment also checked how humans answered the questions. Figure 2.4.1.2 presents the results of the experiment.

	Core Probes			+ Adversarial Examples		
Models	**hit@1**	hit@2	hit@3	**hit@1**	hit@2	hit@3
GPT-2	29.86	50.88	67.49	24.73	44.21	62.30
BERT-Base	31.98	55.92	70.58	25.24	48.66	64.81
RoBERTa-Base	36.04	60.42	72.08	28.39	51.91	67.29
BERT-Large	37.63	62.01	76.77	27.18	52.89	70.22
RoBERTa-Large	**45.85**	66.70	80.04	**35.66**	58.52	74.44
Ft. BERT-L.	50.00	66.34	74.91	43.58	62.27	72.92
Ft. RoBERTa-L.	**54.06**	69.61	79.15	**47.52**	66.43	76.76
Human Bound	$89.7^{(\alpha)}$ / $96.3^{(\beta)}$			$88.3^{(\alpha)}$ / $93.7^{(\beta)}$		

2.4.1.2 Numerical common-sense capabilities (Lin et al., 2020)

Before the results of the experiment will be discussed, certain things must be explained. Ft. stands for fine-tuned. (α) means humans could not use external sources of information. (β) means humans could use Wikipedia.

As it can be seen in figure 2.4.1.2 both GPT-2's and BERT's numerical common-sense capabilities are much below the human's numerical common-sense capabilities. Humans managed to provide a correct answer about almost 90% without using external sources of information and above 90% when using Wikipedia. GPT-2's numerical common-sense capabilities seem to be the lowest from all pre-trained language models used in the experiment. BERT achieved much better results than the GPT-2, however, the RoBERTa achieved the best results from all pre-trained language models used in the experiment.

The conclusion is that both BERT's and GPT-2's numerical common-sense capabilities are still far below the human's numerical common-sense capabilities.

As it was presented above both BERT's and GPT-2's numerical common-sense capabilities are quite low. There is still a lack of accuracy. In another experiment presented in the publication "CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning", the authors focused on pre-trained language models' generative commonsense reasoning capabilities. Commonsense reasoning is "the ability to make acceptable and logical assumptions about ordinary scenes in our daily life" (Lin et al., 2019).

There was created a dataset. For instance, based on the words "(e.g., {dog, frisbee, catch, throw}); the task is to generate a coherent sentence describing an everyday scenario using these concepts (e.g., "a man throws a frisbee and his dog catches it")" (Lin et al., 2019).

Figure 2.4.1.3 presents some results both for humans and machines.

Figure 2.4.1.3 Generative Commonsense Reasoning (Lin et al., 2019)

As t can be seen in figure 2.4.1.3 the results produced by the machines are far below the humans. For instance, GPT-2 produces the sentence “dog throws a frisbee at a football player”, which is far from making any sense.

Figure 2.4.1.4 presents other results produced by the machines and humans for words {hand, sink, wash, soap}

Figure 2.4.1.4 Generative Commonsense Reasoning (Lin et al., 2019)

Same as before some sentences make no sense. For instance, the sentence produced by the BERT pre-trained language model “a woman washes her hands with a sink of soaps” is a bit far from making sense.

Conclusion both BERT (and its variants) and GPT-2 have some common-sense capabilities. However, their common-sense capabilities are far from perfect. Now, the common-sense capabilities of the pre-trained language models are far below the human ones. The pre-trained language models were a huge step forward in AI, but there must be more work done to make pre-trained language models' common-sense capabilities close to the human ones. The time will show if or when it will be achieved.

3 Design and Implementation

This chapter will start by presenting the datasets used for training and evaluating the pre-trained language models (Section 3.1). It will cover the raw datasets, checking and cleaning datasets, pre-processing data for fine-tuning the models, and adding additional context to the datasets for models' evaluation. Then the next section (Section 3.2) will present the pre-trained language models, starting with not fine-tuned models and ending with fine-tuned models.

3.1 Datasets

Two datasets will be used to evaluate the selected pre-trained models and their capabilities of showing common sense. Both datasets can be found under the link:
https://huggingface.co/datasets/cstrathe435/Task2Dial/tree/main
More information about the datasets and the datasets' creators can be found under the link:
https://aclanthology.org/2021.icnlsp-1.28.pdf

As mentioned above two datasets will be used. The first dataset is an alternative products dataset, containing the products and alternative products that can be used as a substitute. The second dataset is a utensils description dataset, containing the utensil name and a short description of the utensil.

3.1.1 Raw datasets

First will be presented the alternative products dataset file. The table below (Table 3.1.1.1) shows some alternative products dataset examples.

Product	Alternative products
if name == "plain flour":	message = "instead you can use; oat flour, bread flour, cake flour or coconut flour, do you have any of these ingredients?"
if name == "olive oil":	message = "instead you can use; peanut oil, butter, coconut milk, ghee, walnut oil, sunflower oil, canola oil or vegetable oil, do you have any of these ingredients?"
if name == "salt":	message = "instead you can use; mint, rosemary, nutmeg, basil, cardamon, chili, cinnamon or chives, do you have any of these ingredients?"
if name == "black pepper":	message = "instead you can use; white pepper, cayenne pepper or papaya seeds, do you have any of these ingredients?"
if name == "eggs":	message = "instead you can use; commercial egg substitute or tofu, do you have any of these ingredients?"

Table 3.1.1.1 Alternative products raw dataset

The alternative products dataset contains 371 products with context containing alternative products that can be used as a substitute.

The second presented dataset file will be a utensils description. The table below (Table 3.1.1.2) shows some utensils description dataset examples.

Utensil	Utensil description
if name == "bowl":	message = "A cooking container that is usually larger than a cup and kept in a cupboard in the kitchen, it is typically made from glass, ceramic, plastic."
if name == "spoon":	message = "an eating or cooking utensil consisting of a small shallow bowl with a relatively long handle made from metal, plastic, wood it is usually kept in the kitchen, in drawer."
if name == "sieve":	message = "A metal or plastic device with meshes or perforations through which finer particles of a mixture are sifted, it is usually kept in a kitchen and stored in a cupboard."
if name == "pan":	message = "A shallow and open cooking container made from metal, it is usually found in a kitchen and stored in a cupboard."
if name == "baking tray":	message = "A metal rectangular sheet with a rolled edge used for baking, it is usually found in a kitchen inside a cupboard or draw."

Table 3.1.1.2 Utensils description raw dataset

Utensil's description dataset contains 202 utensils with corresponding context containing short description of the utensil.

As t can be noticed both dataset files have the same structure. First is the keyword, the name of the product or the name of the utensil. Then is the context, which depending on the dataset file contains the list of alternative products or a short description of the utensil.

3.1.2 Preparing raw datasets

Raw dataset files cannot be pre-processed. Both dataset files must be carefully checked and cleaned and rearranged before they can be pre-processed. This will be two steps process. The first step will be manual checking and cleaning each dataset file to ensure all data is consistent and free of errors, such as structure errors, misspelling words, additional spaces, or unexpected characters that could cause problems in the next step. Once it gets done, the second step will focus on rearranging the structure of the dataset files.

First will be presented the alternative products dataset. The manual checking and cleaning of alternative products dataset resulted in removing one product and corresponding context because the context did not contain any alternative products that could be used as a substitute. Therefore, there was no reason to have a product and context with no alternative substitute. During the manual checking and cleaning, there were found and removed some structure inconsistencies such as additional spaces or characters which could affect the next step, rearranging the data.

Once the manual checking and cleaning were finished the file was saved and uploaded to google drive. Then the file was loaded with the usage of python and the structure of data was rearranged the following way.

- all quotation marks (“) were removed
- all if name == were removed
- all message = were removed
- there was left only product name with semicolon and alternative products that can be used as substitutes with added semicolon, all other text was removed

The table below (Table 3.1.2.1) presents some examples of alternative products dataset after checking, cleaning, and rearranging the data.

Product	Alternative products
plain flour :	oat flour, bread flour, cake flour or coconut flour :
olive oil :	peanut oil, butter, coconut milk, ghee, walnut oil, sunflower oil, canola oil or vegetable oil :
salt :	mint, rosemary, nutmeg, basil, cardamon, chili, cinnamon or chives :
black pepper :	white pepper, cayenne pepper or papaya seeds :
eggs :	commercial egg substitute or tofu :

Table 3.1.2.1 Alternative products cleaned dataset

As it can be seen in table 3.1.2.1, the rearranged alternative products dataset file contains only the product with a semicolon (:) and alternative substitutes with an added semicolon at the end (:). Everything else was removed. Such dataset was saved to google drive and is now ready for pre-processing.

The same steps were taken to prepare the utensils description dataset file. First, the dataset was checked and cleaned manually. This resulted in finding and removing some structure inconsistencies such as additional spaces or characters which could affect the next step, rearranging the data.

Once the manual checking and cleaning were finished the file was saved and uploaded to google drive. Then the file was loaded with the usage of python and the structure of data was rearranged the following way.

- all quotation marks (“) were removed
- all if name == were removed
- all message = were removed
- there was left only utensil name with semicolon and utensil description with added semicolon

Table below (Table 3.1.2.2) presents some examples of utensils description dataset after checking, cleaning, and rearranging the data.

Utensil	Utensil description
bowl :	a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen, it is typically made from glass, ceramic, plastic. :
spoon :	an eating or cooking utensil consisting of a small shallow bowl with a relatively long handle made from metal, plastic, wood it is usually kept in the kitchen, in drawer. :
sieve :	a metal or plastic device with meshes or perforations through which finer particles of a mixture are sifted, it is usually kept in a kitchen and stored in a cupboard. :
pan :	a shallow and open cooking container made from metal, it is usually found in a kitchen and stored in a cupboard. :
baking tray :	a metal rectangular sheet with a rolled edge used for baking, it is usually found in a kitchen inside a cupboard or draw. :

Table 3.1.2.2 Utensils description cleaned dataset

As it can be seen in table 3.1.2.2 the rearranged utensils description dataset file contains only the utensil name with a semicolon (:) and utensil description with an added semicolon at the end (:). Everything else was removed. Such dataset was saved to google drive and is now ready for pre-processing.

3.1.3 Pre-processing datasets

Once the raw datasets get prepared, they can be pre-processed. Pre-processing the dataset will allow using the dataset to fine-tune the models, BERT, and GPT-2. For each of those models, the datasets must be pre-processed differently.

There will be used simple transformer for fine-tuning the BERT model. The simple transformer requires training datasets to have a form of a single list of dictionaries. Each dictionary must have:

- context – context is a paragraph or text from which the question is asked, for instance: “you can use oat flour, bread flour, cake flour or coconut flour if you do not have plain flour”
- qas – this is a list of questions and answers related to the context.

Questions and answers (Q&A) must have a form of dictionaries. Each dictionary in qas must have the following:

- id – id must be unique across the entire dataset and must have a form of a string, for instance, “0001”
- question – question must be a string, for instance: “What can be used instead of plain flour?”
- is_impossible – this is a bool (True or False) and indicates whether the question can be answered based on information from the context
- answers – the list of answers to the question. There must be at least one answer.

Each answer has a form of dictionary and must contain:

- text – the answer to the question in a form of a string, must be a substring of the context, for instance: “oat flour, bread flour, cake flour or coconut flour”
- answer_start – int, indicating the index in the context where the answer to the question starts

Generally, qas can have multiple questions and multiple answers. However, having more questions and more answers will result in more data for fine-tuning. Too much data for fine-tuning can crush the fine-tuning process. The available resources such as RAM and CPU / GPU are also limited. Therefore, for this project, each qas will have only one question and each question will have only one correct answer.

It will always be possible to answer the question based on the information in the context, therefore, is_ impossible will be always False. There will be presented two examples, one for alternative products (Figure 3.1.3.1) and one for utensils description (Figure 3.1.3.2).

```
{'context': ' you can use oat flour, bread flour, cake flour or coconut
flour if you do not have plain flour',
  'qas': [{'answers': [{'answer': 'oat flour, bread flour, cake flour or co-
conut flour',
      'answer_start': 14,
      'text': 'oat flour, bread flour, cake flour or coconut flour'}],
    'id': '00000',
    'is_impossible': False,
      'question': 'What can be used instead of plain flour?'}]},
```

3.1.3.1 Alternative products dataset example

Explanation of example 3.1.3.1:

- context – “you can use oat flour, bread flour, cake flour or coconut flour if you do not have plain flour”
- qas – as mentioned above has only one question and one correct answer
- answers – answer starts at the index of 14 (answer_start = 14) in the context, this is where the text “oat flour, bread flour, cake flour or coconut flour” start in the context
- question – “What can be used instead of plain flour?”
- id = “00000”
- is_impossible – is False, that means it is possible to answer the question based on the information in the context

Below is second example. This time for utensils description.

```
{'context': 'bowl is a cooking container that is usually larger than a cup
and kept in a cupboard in the kitchen, it is typically made from glass, ce-
ramic, plastic..',
  'qas': [{'answer': 'a cooking container that is usually larger than a cup
and kept in a cupboard in the kitchen, it is typically made from glass, ce-
ramic, plastic.',
    'answers': [{'answer': 'a cooking container that is usually larger than
a cup and kept in a cupboard in the kitchen, it is typically made from
glass, ceramic, plastic.',
      'answer_start': 8,
      'text': 'a cooking container that is usually larger than a cup and
kept in a cupboard in the kitchen, it is typically made from glass, ceramic,
plastic.'}],
    'id': '00000',
    'is_impossible': False,
    'question': 'What is  bowl?'}]},
```

3.1.3.2 Utensils description dataset example

Explanation of example 3.1.3.2:

- context – “bowl is a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen, it is typically made from glass, ceramic, plastic.”
- qas – as mentioned above has only one question and one correct answer
- answers – answer starts at the index of 8 (answer_start = 8) in the context, this is where the text “a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen, it is typically made from glass, ceramic, plastic.” start in the context
- question – “What is bowl?”
- id = “00000”
- is_impossible – is False, that means it is possible to answer the question based on the information in the context

Such pre-processed datasets can be used to fine-tune the BERT model.

The dataset structure for GPT-2 model is a bit simpler than for BERT. The dataset structure is divided into blocks of text, each block of text contains:

- <|startoftext|> - indicates the beginning of the text block
- [CONTEXT]: - context is a paragraph or text from which the question is asked, for instance: “If you do not have plain flour you can use instead oat flour, bread flour, cake flour or coconut flour”
- [QUESTION]: - for instance: “What can I use instead of plain flour?”
- [ANSWER]: - for instance: “oat flour, bread flour, cake flour or coconut flour”
- <|endoftext|> indicates the end of the text block

There will be presented two examples, one for alternative products dataset (Figure 3.1.3.3) and one for utensils description dataset ((Figure 3.1.3.4).

```
<|startoftext|>
[CONTEXT]: If you do not have plain flour you can use instead oat flour,
bread flour, cake flour or coconut flour
[QUESTION]: What can I use instead of plain flour?
[ANSWER]: oat flour, bread flour, cake flour or coconut flour
<|endoftext|>
```

Figure 3.1.3.3 Alternative products dataset example

```
<|startoftext|>
[CONTEXT]: bowl is a cooking container that is usually larger than a cup and
kept in a cupboard in the kitchen, it is typically made from glass, ceramic,
plastic.
[QUESTION]: What is bowl?
[ANSWER]: a cooking container that is usually larger than a cup and kept in
a cupboard in the kitchen, it is typically made from glass, ceramic, plas-
tic.
<|endoftext|>
```

Figure 3.1.3.4 Utensils description dataset example

These two examples are self-explanatory, therefore, there is no need to present additional explanation.

3.1.4 Additional context for dataset

The alternative products dataset and utensils description dataset has one drawback. The context is relatively short. Some models such as BERT, or fine-tuned GPT-2 for question answering, answer the question by extracting the answer to the question from the provided context. Too short context makes it too easy for the model to extract the answer to the question and makes it hard to determine the model's capabilities of showing common sense. Therefore, it is a good idea to provide a bit longer context to evaluate the model's capabilities. However, the context must not be too large. Because each model has a limit of data, the number of tokens, that can receive as input. Too small context makes it too easy for the model to answer the question too long context can cause the model to crush. Using one big or two small paragraphs from Wikipedia seems to be the optimal solution. This way the

context will be big enough to challenge the models but not too large so the model will not crush.

For the alternative products, the context for testing and evaluation of the model's capabilities will be a combination of Wikipedia context (one-two paragraph from Wikipedia) and answer from the alternative products dataset context added to the Wikipedia contex. For instance:

- Wikipedia context - "All-purpose, or AP flour, or plain flour is medium in gluten protein content 9.5-11.5% [11] (10-12% from second source[12]) protein content It has adequate protein content for many bread and pizza bases, though bread flour and special 00 grade Italian flour are often preferred for these purposes, respectively, especially, by artisan bakers. Some biscuits are also prepared using this type of flour. Plain refers not only to AP flour's middling gluten content but also to its lack of any added leavening agent (as in self-rising flour)." (Flour - Wikipedia, 2022)
- Dataset answer - "oat flour, bread flour, cake flour or coconut flour."

The dataset answer must be added to the Wikipedia context because the Wikipedia context does not contain the answer to the question. For instance, the presented above Wikipedia context does not contain the answer to the question of what can be used instead of plain flour. Therefore, the answer must be provided so that the models that extract the answer to the question from context can do the job. The answer to the question can be added to the beginning of Wikipedia context, or the end of Wikipedia context or in the middle of Wikipedia context.

For the utensils' description, the context for testing and evaluation of the model's capabilities will be only a Wikipedia context. For example:

- Wikipedia context - "A **bowl** is a round dish or container typically used to prepare and serve food. The interior of a bowl is characteristically shaped like a spherical cap, with the edges and the bottom forming a seamless curve. This makes bowls especially suited for holding liquids and loose food, as the contents of the bowl are naturally concentrated in its center by the force of gravity. The exterior of a bowl is most often round but can be of any shape, including rectangular. The size of bowls varies from small bowls used to hold a single serving of food to large bowls, such as punch bowls or salad bowls, that are often used to hold or store more than one portion of food. There is some overlap between bowls, cups, and plates. Very small bowls, such as the tea bowl, are often called cups, while plates with especially deep wells are often called bowls." (Bowl - Wikipedia, 2022)
- Dataset context – "a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen, it is typically made from glass, ceramic, plastic."

In the case of utensils' description, the Wikipedia context contains the answer to the question. For instance, The Wikipedia context presented above contains the answer to the question what is a bowl? However, the utensils dataset context answer will be displayed separately during the testing and evaluation so that it can be compared with the answer generated by the model.

3.2 Models

There will be used five models to check the pre-trained language models' capabilities of showing common sense. Three, not fine-tuned models: BERT-large, GPT-2 124M and GPT-2 774M and two fine-tuned models: BERT-base and GPT-2 124M. Using both BERT and GPT-2 models, not fine-tuned and fine-tuned models will allow comparing the models' capabilities of showing common sense.

Google Collab will be used for the implementation of this project. Google Collab offers great capabilities and flexibility. However, free access to Google Collab offers limited resources such as RAM and CPU / GPU which has some implications that will be explained.

Using any pre-trained language model in Google Collab requires certain steps to take place. The next two subsections will present the most necessary steps to use each of the pre-trained language models presented above. The steps depend on the implementation. A different implementation may result in different steps.

3.2.1 Not fine-tuned models

This subsection will present steps required to use not fine-tuned models. It will start with BERT-large, then there will be presented GPT-2 124M and GPT-2 774M.

BERT model requires the following steps:

- Install transformers
- Import BERT model and tokenizer
- Assign BERT model and tokenizer to variables
- Import torch
- Create answerQuestion(question, context) function

At this point, the BERT model is ready and can be used to answer questions. To use the BERT model the function answerQuestion(question, context) must be called and the question and context must be provided.

There will be presented two examples of using not fine-tuned BERT model, one for each of the datasets.

```
question = "What can be used instead of plain flour?"
context = "If you do not have plain flour you can use instead oat flour, bread flour, cake flour or coconut flour"
answerQuestion(question, context)
```

```
Answer: oat flour , bread flour , cake flour or coconut flour
```

3.2.1.1 Alternative products BERT answer question example

```
question = "What is a bowl?"
context = "bowl is a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen,"
context = context + "it is typically made from glass, ceramic, plastic"
answerQuestion(question, context)
```

```
Answer: a cooking container
```

3.2.1.2 Utensils description BERT answer question example

Not fine-tuned BERT-large model is ready for checking its capabilities of showing common-sense.

Both GPT-2 124M and GPT-2 774 models require the same steps. There are some differences which include selecting a different model. Therefore, both models will be presented together, and the differences will be shown. Both GPT-2 124M and GPT-2 774M require the following steps to be used:

- Install GPT-2
- Import GPT-2 model
- Download appropriate GPT-2 model and assign it to the variable

For GPT-2 124M model it must be 124M model, and for GPT-2 774M model it must be 774M

- Import datetime and files
- Load appropriate model

For GPT-2 124M model it must be 124M model, and for GPT-2 774M model it must be 774M

- Create generateAnswer(question) function

After those steps both models GPT-2 124M and GPT-2 774 are ready to be used. To use any of those models to generate an answer to the question the function generateAnswer(question) must be called and the question must be passed.

There will be presented two examples for each model (GPT-2 124M and GPT-2 774M), one for each dataset.

```
question = "What can be used instead of plain flour?"
generateAnswer(question)
```

```
What can be used instead of plain flour?

A lot of people don't need a lot of flour in their food.

You should probably only use flour that does not contain gluten-containing ingredients that taste like wheat.

There are different types of flour and many different kinds
```

3.2.1.3 Alternative products GPT-2 124M answer question example

```
question = "What can be used instead of plain flour?"
generateAnswer(question)
```

```
What can be used instead of plain flour?

Purple flour is high in protein, and can be used instead of plain flour as a flour substitute.

Purple flour can also
```

3.2.1.4 Alternative products GPT-2 774M answer question example

```
question = "What is a bowl?"
generateAnswer(question)
```

```
What is a bowl? How does one make the bowl?

A bowl is a set of vessels which hold a portion of food or drink in one hand and is regarded as a
```

3.2.1.5 Utensils description GPT-2 124M answer question example

```
question = "What is a bowl?"
generateAnswer(question)
```

```
What is a bowl? It is a container that holds the contents of a bowl or mug.

Typically a bowl is used to hold a drink, such as a drink of water, or a drink of tea.

How to Use a Bowl

You can
```

3.2.1.6 Utensils description GPT-2 774M answer question example

All, not fine-tuned models, BERT-large, GPT-2 124M and GPT-2 774M, are ready to be used. It is important to mention that not fine-tuned BERT model uses context to extract the answer, but the GPT-2 models 124M and 774M do not use context, instead, GPT-2 models generate the answer.

3.2.2 Fine-tuned models

This subsection will present steps required to use fine-tuned models. It will start with BERT-base, then there will be presented GPT-2 124M. As mentioned above free access to Google Collab offers limited resources such as RAM and CPU / GPU. Therefore, the GPT-2 774M will not be fine-tuned, because the available resources are insufficient.

Fine-tuning BERT model requires the following steps:

- Install simpletransformer
- Add GPU support
- Import logging, QuestionAnsweringModel and QuestionAnsweringArgs
- Specify model type and model name
- Specify training arguments
- Download the model
- Prepare training data
- Train the model

The model must be trained twice once for each of the datasets. This is because the model has a limit on data, the number of tokens, that can be used during the fine-tuning and because the Google Collab resources are limited. Using both datasets at the same time could result in crushing during the fine-tuning. Therefore, after training the model for the alternative products the model can be used for an alternative products dataset. After getting the results for the alternative products, the model must be fine-tuned again for the utensils'

description, then it can be used to get results for the utensils' description. All models are tested using the same questions with no modifications so the result achieved by the models can be compared.

There will be presented two examples, one for each dataset.

```
question = "What can be used instead of plain flour?"
context = "If you do not have plain flour you can use instead oat flour, bread flour, cake flo
to_predict = [{"context": context, "qas": [{"question": question, "id": "0"}]}]
answers, probabilities = model.predict(to_predict)
print(answers)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 410.40it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 9686.61it/s]
Running Prediction: 100% ████████████████████████████████ 1/1 [00:00<00:00, 14.01it/s]
[{'id': '0', 'answer': ['oat flour, bread flour, cake flour or coconut flour', 'oat flour, bre
```

3.2.2.1 Alternative products fine-tuned BERT answer question example

```
question = "What is a bowl?"
context = "bowl is a cooking container that is usually larger than a cup and kept in a cupboard in
context = context +"it is typically made from glass, ceramic, plastic"
to_predict = [{"context": context, "qas": [{"question": question, "id": "0"}]}]
answers, probabilities = model.predict(to_predict)
print(answers)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 212.43it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 8612.53it/s]
Running Prediction: 100% ████████████████████████████████ 1/1 [00:00<00:00, 16.89it/s]
[{'id': '0', 'answer': ['a cooking container that is usually larger than a cup and kept in a cupboa
```

3.2.2.2 Utensils description fine-tuned BERT answer question example

Steps required to use fine-tuned GPT-2 124M model can be divided into two parts. First, the model must be fine-tuned. Second, the model must be loaded and used.

Fine-tuning GPT-2 124M model requires the following steps:

- Install GPT-2
- Add GPU support
- Import GPT-2 model
- Import datetime and files
- Download GPT-2 model

- Specify file name (training dataset file) and model’s run name
- Mount google drive and copy the data file to the model
- Train the model
- Copy the model checkpoint to google drive

For the same reasons as BERT, the GPT-2 124M model must be trained twice. Once for each dataset, alternative products, and utensils' description. Each time the model gets trained the checkpoint gets copied to google drive. After the model gets fine-tuned, the runtime must be restarted.

After training the model, the runtime must be restarted, and some steps must run again. After restarting the Runtime, the model can be used.

Using fine-tuned GPT-2 124M model requires the following steps:

- Install GPT-2 model
- Import GPT-2 model
- Import datetime and files
- Specify model’s run name

  There are two checkpoints one for the alternative products dataset and one for the utensils’ description dataset. The run name specifies which checkpoint will be copied.
- Mount google drive
- Copy GPT-2 checkpoint from google drive
- Load GPT-2 model
- Create generateAnswer(pre) function

Once the model gets loaded, it can be used. To use the model to answer the question the function generateAnswer(pre) must be called and the pre parameter must be provided. The pre parameter must contain the question and the context.

There will be presented two examples one for each dataset.

```
question = "What can be used instead of plain flour?"
context = "If you do not have plain flour then you can use instead oat flour, bread flour, cake flour or coconut flour."
pre = "\n[CONTEXT]: " + context + "\n[QUESTION]:" + question + "\n[ANSWER]:"
generateAnswer(pre)
```

```
[CONTEXT]: If you do not have plain flour then you can use instead oat flour, bread flour, cake flour or coconut flour.
[QUESTION]:What can be used instead of plain flour?
[ANSWER]: oat flour, bread flour, cake flour or coconut flour.
<|endoftext|
```

3.2.2.3 Alternative products fine-tuned GPT-2 124M model answer question example

```
question = "What is a bowl?"
context = "A bowl is a cooking container that is usually larger than a cup and kept in a cupboard in the
pre = "\n[CONTEXT]: " + context + "\n[QUESTION]:" + question + "\n[ANSWER]:"
generateAnswer(pre)
```

```
[CONTEXT]: A bowl is a cooking container that is usually larger than a cup and kept in a cupboard in the k
[QUESTION]:What is a bowl?
[ANSWER]: a cooking container that is usually larger than a cup and kept in a cupboard in the kitchen,
```

3.2.2.4 Utensils description fine-tuned GPT-2 124M model answer question example

4 Testing and Results

This chapter will present the models' capabilities of showing common sense. The chapter will start with an explanation of the testing method used to check models' capabilities and with an overview of the results achieved by all the models (Section 4.1). Then the next section (Section 4.2) will provide more details about the results achieved by each model.

4.1 Overview of models' results

As mentioned above, there are five models used for this project. Three of them are models without fine-tuning which are BERT-large, GPT-2 124M and GPT-2 774M. Two models, BERT-base, and GPT-2 124M were fine-tuned. As mentioned above GPT-2 774M model could not be fine-tuned because of insufficient resources, RAM and CPU / GPU.

Using both BERT and GPT-2 models, not fine-tuned and fine-tuned models allows comparing not only BERT with GPT-2 but also not fine-tuned models with fine-tuned models. It is important to notice that not fine-tuned BERT is large while the BERT model used for fine-tuning was base.

There is one big problem with using different models. The problem is that different models work differently. Understanding the differences between the models is crucial because they show that different models can achieve different results based on the task they are used for.

All five models presented above can be divided into three groups based on the way they answer the questions. The first group are the not fine-tuned and fine-tuned BERT models. These models answer the questions by extracting the answer to the question from the context. These models do not answer the question themselves; they find the answer to the question in the context. The second group are the models that do not use the context. This group belong not fine-tuned GPT-2 124M and GPT-2 774M models. These models as

mentioned do not use the context, instead, they generate the answers to the questions themselves. Finally, the third group is the model that does use the context to answer the questions, however, it can generate the answer itself. To this group belongs the fine-tuned GPT-2 124M model. This model can be considered a hybrid because it has the capabilities of both groups, it can use the context to extract the answer from it and it can answer the question by generating the answer.

BERT models answer questions by searching the context. Therefore, their capabilities of answering the questions and showing common sense heavily depend on the context. If the context is not provided the quality of generated answers by the BERT models can be very poor. These models can be used for working with text, where the task is to find the answer in the text. These models can be considered as someone who can read and write but does not have the knowledge, so all he or she can do is find the answer to the question in the text.

The second group as mentioned do not use the context. So, the not fine-tuned GPT-2 124M and GPT-2 774M models heavily depend on the knowledge each of these models has. If the model gets a question beyond its knowledge, then the model will struggle to give the correct answer. These models can be compared to someone who cannot read and write, but instead has some knowledge and can speak, therefore, can answer the questions if he has sufficient knowledge.

Finally, the third group combines the benefits of both. It can extract the answer from the text, or it can generate it. The fine-tuned GPT-2 124M model can be compared to someone who can both read and write, therefore can work with text, but can also speak and has some knowledge, therefore can answer the question by generating the answer.

The question that appears is how to classify the answers generated by different models. How to decide whether the answer is correct or not. Two criteria will be used to classify the answers. The first criteria will be whether the answer matches the expected answer in the context. The second criteria will be whether the answer makes sense.

Each answer generated by the model will be classified to one of the five categories.

- Correct answers – if the whole answer match the expected answer from the context or if the whole answer makes sense. Combination of both criteria will also be considered as correct answer.
- Partially correct answers – if part of the answer match the expected answer from the context or if part of the answer makes sense. Combination of both criteria will also be considered as partially correct answer.
- General answers – if the answer makes sense but lacks details, or if the answer match small part of the expected answer from the context but it lacks details. Combination of both criteria will also be considered as partially correct answer.
- Wrong answers – if the answer does not make sense at all or if the answer is completely different from the expected answer in the context.
- No answers – if the model does not answer the question or if the answer will be “empty”.

The models’ capabilities were tested the following way. Each model was first used to answer 30 questions related to the alternative products. For instance, each model was used to answer the question: “What can be used instead of plain flour?” The answers generated by the models were classified based on the criteria explained above.

Then each model was used again to answer 30 questions, this time each model was asked to describe a kitchen utensil. For instance, each model was used to answer the question: “What is a bowl?” The answers generated by the models were classified based on the criteria explained above.

Table 4.1.1 presents the results achieved by the models when answering questions related to the alternative products.

Model	Correct answers	Partially correct answers	General answer	Wrong answers	No answers
Not fine-tuned BERT-large	83.33%	0%	16.66%	0%	0%
Not fine-tuned GPT-2 124M	23.33%	0%	76.66%	0%	0%
Not fine-tuned GPT-2 774M	56,66%	0%	43,33%	0%	0%
Fine-tuned BERT-base (*)	100% 30%	0% 3,33%	0% 26,66%	0% 13,33%	0% 26,66%
Fine-tuned GPT-2 124M	100%	0%	0%	0%	0%

Table 4.1.1 Alternative products – models’ results

Table 4.1.2 presents the results achieved by the models when answering the questions related to kitchen utensils.

Model	Correct answers	Partially correct answers	General answer	Wrong answers	No answers
Not fine-tuned BERT-large	66,66%	0%	33,33%	0%	0%
Not fine-tuned GPT-2 124M	56,66%	16,66%	10%	16,66%	0%
Not fine-tuned GPT-2 774M	63,33%	16,66%	10%	10%	0%
Fine-tuned BERT-base	76,66%	0%	3,33%	0%	20%
Fine-tuned GPT-2 124M	90%	3,33%	0%	6,66%	0%

Table 4.1.2 Utensils’ description – models’ results

(*) – for fine-tuned BERT for alternative products there are two numbers for each column. The first number is the results the model achieved when the answer to the question was at the beginning of the context. The second number is the results the model achieved when the answer to the question was at the end of the context.

4.2 Details of models' results

This section will present details of the models' results. First will be presented results of not fine-tuned models, then there will be presented results of the fine-tuned models.

4.2.1 Not fine-tuned BERT

As it was presented in Table 4.1.1, the not fine-tuned BERT model managed to answer correctly 83.33% of questions related to the alternative products. 23 of those answers generated by the BERT model were full answers, BERT found all alternative products that were in the context. 2 of those answers were partial answers, and BERT found some of the alternative products from the context. 16.66% of answers generated by the BERT model were general, BERT did not manage to find alternative products, the answer given by BERT to those 16.66% questions was "other products", which is not the wrong answer. Therefore, it was classified as a general answer.

As it was presented in Table 4.1.2, the not fine-tuned BERT model managed to answer correctly 66,66% of questions related to the kitchen utensils. All correct answers were good descriptions of kitchen utensils. 33,33% of answers given by the BERT model were classified as general answers. They were not wrong answers, they just lack details. The answers in this category were for instance: "utensil", "kitchen utensil", "pan" etc. They were short answers without any details. Therefore, they were classified as general answers.

Not fine-tune BERT showed it can understand the text and based on the provided text, the context, it can answer the question. BERT showed it has some capabilities of showing common sense.

4.2.2 Not fine-tuned GPT-2 124M

As it was presented in Table 4.1.1, the not fine-tuned GPT-2 124M model managed to correctly answer 23.33% of the questions. Those answers were classified as correct because the model specified at least one product that can be used as a substitute. And as was mentioned above without domain-specific knowledge the model can't provide the answer that will match the expected answer from the context. 76.66% of answers generated by the model were general answers where the model gave some information about the product but did not specify any alternatives.

As it was presented in Table 4.1.2, the not fine-tuned GPT-2 124M model managed to answer correctly 56,66% of questions related to the kitchen utensils. All correct answers were good descriptions of kitchen utensils. 16,66% of answers were classified as partially correct answers. Part of the answer was correct descriptions of the kitchen utensil part of the answer did not make sense at all. 10% of answers were classified as general, these answers lacked any details to be classified as correct or wrong answers. 16,66% of answers were classified as wrong answers, these answers make no sense at all.

Not fine-tune GPT-2 124M model showed it can understand the question and based on its knowledge it can generate the answer to the question that makes sense. GPT-2 124M showed it has some capabilities of showing common sense.

4.2.3 Not fine-tuned GPT-2 774M

As it was presented in Table 4.1.1, the not fine-tuned GPT-2 774M model managed to correctly answer 56,66% of the questions. Each of those answers was classified as correct because the model specified at least one product that can be used as an alternative. Same as with not fine-tuned GPT-2 124M model, without domain-specific knowledge, the model cannot provide the answer that will match the expected answer from the context.

43,33% of answers generated by the model were general answers, the model gave some information about the product but did not specify any substitutes.

As it was presented in Table 4.1.2, the not fine-tuned GPT-2 774M model managed to answer correctly 63,33% of questions related to the kitchen utensils. Each of these answers was a good description of the kitchen utensil. 16,66% of answers were classified as partially correct answers. Part of the answer was correct descriptions of the kitchen utensil part of the answer did not make sense at all. 10% of answers were classified as general, these answers lacked any details to be classified as correct or wrong answers. 10% answers were classified as wrong answers, these answers make no sense at all.

Same as not fine-tuned GPT-2 124M, not fine-tune GPT-2 774M model showed it can understand the question and based on its knowledge it can generate the answer to the question that makes sense. GPT-2 774M showed it has some capabilities of showing common sense.

4.2.4 Fine-tuned BERT

Fine-tuning the BERT was not fully successful. There was used simpletransformers instead of transformers and BERT-base instead of BERT-large. The selected training arguments and the whole process managed to run without any errors, however, the model's capabilities of showing common sense changed unpredictably. The BERT model still can answer the questions by searching the context to find the answer. However, the way the context is structured makes a huge impact on BERT's performance. If the answer to the question is at the beginning of the context, then the fine-tuned BERT finds it easily. If the answer to the question is at the end of the context, then the BERT model struggles to find it. Maybe it is because BERT-base was used instead of BERT-large. It can be specially noticed when asking fine-tuned BERT questions related to alternative products. Therefore, for

alternative products there are two results in each column, showing different results based on where the answer to the question was placed in the context.

As it was presented in Table 4.1.1, the fine-tuned BERT model managed to correctly answer 100% of questions if the answer to the question was at the beginning of the context. When the answer to the question was at the end of the context the model managed to answer correctly only 30% questions. 3.33% of the questions were answered partially correct. 26.66% of answers generated by the fine-tuned BERT model were general, and 13.33% of answers were wrong. Fine-tuned BERT did not manage to answer 26.66% of questions when the answer to the question was at the end of the context. The model answer was “empty”. As it can be seen placing the answer to the question at the beginning of the context significantly improvise the models’ capabilities of finding it. On the other hand, when the answer to the question is at the end the model struggles to find it. Answers generated by the model were classified the same way they were classified in the case of the not fine-tuned BERT model.

As it was presented in Table 4.1.2, the fine-tuned BERT model managed to answer correctly 76,66% of questions related to the kitchen utensils correctly. 3.33% of the answers generated by the BERT was general answer. Fine-tuned BERT did not manage to answer 20% of questions, the answer generated by BERT was “empty”.

Fine-tuned BERT proved it has some capabilities of showing common-sense. However, as it was mentioned the process of fine-tuning BERT was not fully successful. The model behaves more unpredictably after fine-tuning.

4.2.5 Fine-tuned GPT-2 124M

Fine-tuning the GPT-2 124M model allowed the model to gain new capability. After fine-tuning the model can answer the questions by searching the context for the answer. If it fails, the model can still generate the answer itself based on the knowledge the model has. As mentioned above, the model can be considered as a hybrid. Those new capabilities of the model affected the results the model achieved.

As it was presented in Table 4.1.1, the fine-tuned GPT-2 124M model managed to correctly answer 100% questions. Those answers were classified as correct because the model managed to find or generate at least one product that can be used as a substitute. Most of the answers match the expected answers from the context. However, there are some examples of the answers, where the fine-tuned GPT-2 model generate completely different products that can be used as a substitute.

As it was presented in Table 4.1.2, the fine-tuned GPT-2 124M model managed to answer correctly 90% questions related to the kitchen utensils. 90% answers were classified as correct answers because they match the expected answers in the context or they make sense, 3,33% answers generated by the model were classified as partially correct answers, 6,66% answers were classified as wrong answers.

Fine-tune GPT-2 124M model showed has good capabilities of showing common sense. Combining capabilities of extracting answers from the context with generating the answers resulted in good answers generated by the model.

5. Analysis and Discussion

This chapter will present the analysis and discussion of the results achieved by the models. The chapter will start with an analysis of the models' results (Section 5.1). Then the chapter will discuss and compare the models' capabilities of showing common sense.

5.1 Analysis

As was presented above each of the models presented some capabilities of showing common sense. However, the capabilities of showing common sense vary for different models.

The task that was used to test and evaluate the models' capabilities was answering the questions where the context was provided for the models that use the context. **This was the perfect task for the BERT models and fine-tuned GPT-2 model.** The GPT-2 model will be analysed a bit later as it is a hybrid, as mentioned above. As was mentioned BERT models can work with the text. These models can extract the answers from the text.

The not fine-tuned BERT model showed it has some common-sense capabilities as it can understand the question and find the answer to the question in the text. Understanding the question and the text, the context, requires some common sense, so the model understands what it is asked for and what the answer should be. This requires understanding the contextual meaning of the words. Understanding the question and the context allows the BERT to answer the question by finding the answer to the question in the context, extracting the answer from the context. Otherwise, the model could not understand what is asked for and what the correct answer to the question should be. Results achieved by not fine-tuned BERT are pretty good. However, as was mentioned above the results the not fine-tuned BERT model achieved are the consequence of selecting this type of task for testing and evaluating the models.

The fine-tuned BERT model on the one hand outperformed the not fine-tuned BERT on the other hand underperformed the not fine-tuned BERT depending on the structure of the context. As mentioned above using the simpletransformers instead of transformers and the BERT-base instead of the BERT-large could affect the way the fine-tuned BERT behaves and performs. Same as the not fine-tuned BERT, the fine-tuned BERT can understand the question and can find the answer to the question in the text, the context. However, as mentioned above on the one hand the fine-tuned BERT showed great performance, when the answer to the question was at the beginning of the context. But, on the other hand, the fine-tuned BERT struggled to answer the questions when the answer to the question was at the end of the context. Nevertheless, the fine-tuned BERT showed it has some capabilities of showing common sense.

Not a perfect task for the not fine-tuned GPT-2 models, nevertheless, both GPT-2 124M and GPT-2 774M showed they have some capabilities of showing common sense. As mentioned above, this task was perfect for the models that use the context to answer the questions. The not fine-tuned GPT-2 models answer the question by generating the answer as mentioned above. So, these models did not benefit from this task. However, both the GPT-2 124M and GPT-2 774M managed to answer the questions. Both models showed they can understand the questions and they can generate the answers based on the knowledge they have. This requires some common sense. Therefore, both models showed they have some capabilities of showing common sense. The 774M model seems to have a bit better capability than the 124M model, which is natural as it is more powerful.

Thanks to new skill the fine-tuned GPT-2 124M outperformed other models. After fine-tuning, the GPT-2 124M model learned how to use the context to answer the questions. And as mentioned above the fine-tuned GPT-2 used both the capability to generate the answer to the question and the capability to find the answer to the question in

the context. Combining both capabilities allowed the fine-tuned GPT-2 124M to achieve the best results from all models. For most questions, the GPT-2 found the answer to the question in the context. However, a few times the fine-tuned GPT-2 124M models used its capabilities to generate the answer as an alternative way to answer the question. The fine-tuned GPT-2 124M proved it has capabilities of showing common sense.

5.2 Discussion

As mentioned above all models have some capabilities of showing common sense. The question that appears is which of the presented models has the best capabilities. It seems that the fine-tuned GPT-2 model, which outperformed all other models, showed the best capabilities of showing common sense. But before the fine-tuned GPT-2 model will be discussed, it is a good idea to discuss other models for comparison.

Not fine-tuned BERT model has solid capabilities of showing common sense. As it was shown above this model can read and understand the question and the context and can answer the question by finding the answer in the context. This model managed to answer each question. Most of the answers generated by this model were correct answers. The model managed to answer correctly for 25 out of 30 questions related to alternative products which are 83.33% of correct answers and 20 out of 30 questions related to utensils' description which are 66.66% of correct answers. This model managed to answer correctly for 45 out of 60 questions which give this model 75% of correct answers. The rest were general answers, which were not wrong, they just lacked details. This is a very good result. However, this model did not achieve the best results.

Fine-tuned BERT model showed it has solid capabilities of showing common sense. However, as mentioned above the capabilities of this model are heavily dependent on the structure of the context. On the one hand, this model managed to answer correctly all

questions related to the alternative products when the answer to the question was at the beginning. On the other hand, this model did not manage to answer all questions related to the utensils' description. When the answer to the question was at the end of the context, the answers generated by the model for alternative products were worse than not fine-tuned BERT model. This model did not achieve the best results.

Both not fine-tuned GPT-2 124M and GPT-2 774M showed that they have some capabilities of showing common sense. Each of these models managed to answer each question. However, the lack of domain-specific knowledge resulted in a huge variety of answers. These models, especially GPT-2 124M, struggled to specify the alternative product that can be used as a substitute. The fact, that these models do not use the context to answer the questions combined with the lack of domain-specific knowledge resulted in good but not the best results achieved by models.

Fine-tuned GPT-2 124M achieved the best results. This model answered correctly 30 out of 30 questions related to the alternative products and 27 out of 30 questions related to the utensils' description. This gives this model the best result, 95% of correct answers. The capability of extracting the answers from the context combined with the capability of generating the answers helped this model achieve the best results.

Even though fine-tuned GPT-2 124M model showed very good capabilities of showing common sense it is important to mention that the testing included only 60 questions. The sample was relatively small and there were used only two datasets. Therefore, there must be done more researching. Now the capabilities of showing the common sense of the presented models are far from the human capabilities of showing common sense. For instance, the human can answer questions from different domains. The fine-tuned GPT-2 124M model requires switching the checkpoint. Human capabilities of showing common sense are superior to the computers at the moment.

6. Conclusion

This chapter will start with critical analysis (Section 6.1). Then there will be presented suggestions regarding the work that should be done in the future (Section 6.2). This chapter will end with presentation of the results (Section 6.3).

6.1 Critical analysis

The models' capabilities of showing common sense were tested with the usage of a very small sample. Each model was asked to answer only 30 questions related to the alternative products and 30 questions related to the utensils' description. Such a small sample used for testing and evaluation has some limitations. It is possible that if the models were used to answer more questions related to the alternative products and the utensils' description, the results achieved by the model would be completely different. Therefore, the results of the models cannot be considered to be final. More testing is required.

The fine-tuned models, BERT and GPT-2 124M, were compared with not fine-tuned models. However, the comparison is not fair. Because for each task there was used different fine-tuned model. One fine-tuned BERT model was used to answer the questions related to the alternative products. Another fine-tuned model was used to answer questions related to the utensils' description. The same with GPT-2. One fine-tuned GPT-2 124M model was used to answer questions related to the alternative products. Another fine-tuned GPT-2 124M was used to answer questions related to the utensils' description. Two fine-tuned models can achieve better results than one not fine-tuned model.

For fine-tuning the BERT, there was used as it was mentioned above simpletransformers instead of transformer and BERT-base instead of BERT-large. Using transformers and modifying the fine-tuning process could result in better results being achieved by the fine-tuned BERT models. Also using BERT-large, which is much more powerful than BERT-base, could result in better performance of the fine-tuned model.

6.2 Further work

As it was stated in the previous sections (Section 6.1) more testing is required before it will be possible to consider the results achieved by the models to be final. Using all questions related to the alternative products and utensils' description would be the ideal situation. Such testing would allow ensuring that the results achieved by the models are final results that show real models' capabilities of showing common sense.

It would be a good idea to fine-tune both BERT and GPT-2 124M models at the same time for both datasets. This task could be very difficult to be accomplished because such a task would require a significant number of resources. Also, each model has some limitations of data, tokens, that can be used during fine-tuning the process. However, if it would be possible to fine-tune both BERT and GPT-2 124M at the same time for both datasets, then the comparison between the not fine-tuned models and the fine-tuned models would be fairer. There would be one model used for both datasets. Also fine-tuning the GPT-2 774M would be a great step forward. It is probable that the fine-tuned GPT-2 774M would outperform other models and would present the best capabilities of showing common sense.

Fine-tuning the BERT model once again could allow the model to achieve better results. As it was mentioned above, the fine-tuning of the BERT model did not go well. The process did not crash but the model started behaving a bit weird. The model's capabilities of showing common sense depending on the structure of the context. Using the transformers instead simpletransformers and optimizing the fine-tuning process could result in a much better model. Also using the BERT-large instead of BERT-base could result in better performance of the model. BERT-large model, which was used in case of not fine-tuned BERT, can be the reason for the better performance of the not fine-tuned BERT model when the answer to the question is at the end of the context.

Using a larger dataset could help to better determine the models' capabilities of showing common sense. Also using datasets in similar domains can be considered an option to help determine the models' capabilities of showing common sense.

6.3 Results

The results presented in the report can be considered satisfactory. Not everything went the way it was expected. Fine-tuning the BERT model did not go well. The process was successful, but the model started behaving a bit weird. Suing simpletransformers instead of transformer and BERT-base instead of BERT-large can be the reason.

As mentioned, there were created and used five pre-trained language models. Each model that was created and used in this project showed that it has some capabilities of showing common sense. The fine-tuned GPT-2 124M model managed to achieve the best results from all five models, as mentioned above. Nevertheless, other models showed that they also have huge potential.

Results achieved by the models can be considered very good. These results prove that the pre-trained language models such as BERT and GPT-2 have some capabilities of showing common sense. Further research is required. However, with more work and testing the models can have very good capabilities of showing common sense. The results achieved by the models give hope that soon the pre-trained language models such as BERT and GPT-2 will have capabilities of showing common sense that will be comparable to humans. The future seems to be bright

References

Alsultanny, Y. A., & Aqel, M. M. (2003). Pattern recognition using multilayer neural-genetic algorithm. Neurocomputing (Amsterdam), 51, 237–247. https://doi.org/10.1016/S0925-2312(02)00619-7

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). Recurrent neural networks for short-term load forecasting : an overview and comparative analysis . Springer. https://doi.org/10.1007/978-3-319-70338-1

Chen, L., Tan, X., Wang, D., Zhong, F., Liu, X., Yang, T., Luo, X., Chen, K., Jiang, H., & Zheng, M. (2020). TransformerCPI: improving compound-protein interaction prediction by sequence-based deep learning with self-attention mechanism and label reversal experiments. Bioinformatics (Oxford, England), 36(16), 4406–4414. https://doi.org/10.1093/bioinformatics/btaa524

Cheng, R. (2020, July 22). Fine-tuning GPT2 for Text Generation Using Pytorch. Retrieved December 31, 2020, from https://towardsdatascience.com/fine-tuning-gpt2-for-text-generation-using-pytorch-2ee61 a4f1ba7

Chernyavskiy, A., Ilvovsky, D., & Nakov, P. (2021). Transformers: “The End of History” for NLP?

Chiang, Huang, S.-F., & Lee, H. (2020). Pretrained Language Model Embryology: The Birth of ALBERT.

Clark, J., 2021. GPT-2: 6-month follow-up. *OpenAI*. Available at: https://openai.com/blog/gpt-2-6-month-follow-up/ [Accessed November 15, 2021].

Clinciu, Gkatzia, D., & Mahamood, S. (2021). It’s Common Sense, isn’t it? Demystifying Human Evaluations in Commonsense-enhanced NLG systems.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT’19). pp. 4171-4186. Minneapolis, MN, USA (2019)

Dev, S., Hassan, S., & Phillips, J. M. (2021). Closed form word embedding alignment. *Knowledge and Information Systems*, *63*(3), 565–588. https://doi.org/10.1007/s10115-020-01531-7

En.wikipedia.org. 2022. *Bowl - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Bowl> [Accessed 12 February 2022].

En.wikipedia.org. 2022. *Flour - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Flour#Plain_or_all-purpose_flour> [Accessed 12 February 2022].

Geetha, M. ., & Karthika Renuka, D. (2021). Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model. *International Journal of Intelligent Networks*, *2*, 64–69. https://doi.org/10.1016/j.ijin.2021.06.005

Han, H., Li, Y., & Zhu, X. (2019). Convolutional neural network learning for generic data classification. Information Sciences, 477, 448–465. https://doi.org/10.1016/j.ins.2018.10.053

Jang, M., & Kang, P. (2020). Paraphrase thought: Sentence embedding module imitating human language recognition. Information Sciences, 541, 123–135. https://doi.org/10.1016/j.ins.2020.05.129

Kieuvongngam, Tan, B., & Niu, Y. (2020). Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2.

Kim, D., Seo, D., Cho, S., & Kang, P. (2019). Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. Information Sciences, 477, 15–29. https://doi.org/10.1016/j.ins.2018.10.006

Kocmi, T., & Bojar, O. (2018). SubGram: Extending Skip-gram Word Representation with Substrings. https://doi.org/10.1007/978-3-319-45510-5_21

Kumar, N., Kumar, S., Dev, A., & Naorem, S. (2021). Leveraging Universal Sentence Encoder to Predict Movie Genre. 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 1, 1013–1018. https://doi.org/10.1109/ICACCS51430.2021.9441685

Lan, Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.

Liao, Zeng, B., Yin, X., & Wei, P. (2020). An improved aspect-category sentiment analysis model for text sentiment analysis based on RoBERTa. Applied Intelligence (Dordrecht, Netherlands), 51(6), 3522–3533. https://doi.org/10.1007/s10489-020-01964-1

Li, Q., Yao, N., Zhao, J., & Zhang, Y. (2021). Self attention mechanism of bidirectional information enhancement. Applied Intelligence (Dordrecht, Netherlands). https://doi.org/10.1007/s10489-021-02492-2

Lin, Lee, S., Khanna, R., & Ren, X. (2020). Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-trained Language Models.

Lin, Zhou, W., Shen, M., Zhou, P., Bhagavatula, C., Choi, Y., & Ren, X. (2019). CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning.

Liu, Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating Wikipedia by Summarizing Long Sequences.

Liu, Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2021). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. Adv. Neural Inf. Process. Syst. 2013, 26, 3111–3119.

Nadkarni, P., Ohno-Machado, L. and Chapman, W., 2021. *Natural language processing: an introduction*.

Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: EMNLP

Sabharwal, N. and Agrawal, A., 2021. *Hands-on Question Answering Systems with BERT*. p.16.

Sabharwal, N. and Agrawal, A., 2021. *Hands-on Question Answering Systems with BERT*. p.20.

Sabharwal, N. and Agrawal, A., 2021. *Hands-on Question Answering Systems with BERT*. p.33.

Sabharwal, N. and Agrawal, A., 2021. *Hands-on Question Answering Systems with BERT*. p.35.

Solaiman, I., 2021. GPT-2: 1.5B release. *OpenAI*. Available at: https://openai.com/blog/gpt-2-1-5b-release/ [Accessed November 15, 2021].

Song, H.-J., Yoon, B.-H., Youn, Y.-S., Park, C.-Y., Kim, J.-D., & Kim, Y.-S. (2018). A method of inferring the relationship between Biomedical entities through correlation analysis on text. Biomedical Engineering Online, 17(Suppl 2), 155–155. https://doi.org/10.1186/s12938-018-0583-4

Strathearn, & Gkatzia, D. (2021). The Task2Dial Dataset: A Novel Dataset for Commonsense-enhanced Task-based Dialogue Grounded in Documents.

Sun, Y., Zheng, Y., Hao, C., & Qiu, H. (2021). NSP-BERT: A Prompt-based Zero-Shot Learner Through an Original Pre-training Task--Next Sentence Prediction.

Radford, A., 2021. Better language models and their implications. *OpenAI*. Available at: https://openai.com/blog/better-language-models/#fn1 [Accessed November 5, 2021].

Terena Bell and Thor Olavsrud. (2020). What is natural language processing? The business benefits of NLP explained. CIO.

Terena Bell and Thor Olavsrud. (2021). What is NLP? Natural language processing explained. CIO.

Tseng, S.-Y., Georgiou, P., & Narayanan, S. (2019). Multimodal Embeddings from Language Models.

Tu, L., Gimpel, K., & Livescu, K. (2017). Learning to Embed Words in Context for Syntactic Tasks.

Turney, P. D. and Pantel, P. From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research, 37:141–188, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.

Wang, Y.-A., & Chen, Y.-N. (2020). What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding.

Westland, S. (1998). Artificial neural networks explained, pt. 1. Journal of the Society of Dyers and Colourists, 114(10), 274–276.

Whitfield. (2021). Using GPT-2 to Create Synthetic Data to Improve the Prediction Performance of NLP Machine Learning Classification Models.

Yamaguchi, A., Chrysostomou, G., Margatina, K., & Aletras, N. (2021). Frustratingly Simple Pretraining Alternatives to Masked Language Modeling.

Zhou, Y., Li, D., Huo, S., & Kung, S.-Y. (2021). Shape autotuning activation function. Expert Systems with Applications, 171, 114534–. https://doi.org/10.1016/j.eswa.2020.114534

Ziegler, D., 2020. Fine-tuning GPT-2 from human preferences. *OpenAI*. Available at: https://openai.com/blog/fine-tuning-gpt-2/ [Accessed November 15, 2021].

Appendix A: IPO

Jerzy Bajer 40428545

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project: Spoken Dialogue Systems / Human-Robot Interaction Applications

Overview of Project Content and Milestones

The goal of this project is to research, compare and test selected pre trained language models for their capabilities of showing common-sense. The project can be divided into two main parts. Firstly is the research. The first task is researching and understanding what the pre trained language models are and how they work. Then is the research of different pre trained language models, their strengths and weaknesses and comparing them based on the research. Secondly is the testing of selected pre trained language models. First is the research how to test selected pre trained language models. Then is testing and comparing the selected pre trained language models.

The Main Deliverable(s):

The research, comparison and test of selected pre trained language models. The research will be in a form of literature review. The comparison and test will include coding.

The Target Audience for the Deliverable(s):

Researchers, students and other people interested in pre trained language models.

The Work to be Undertaken:

Researching and understanding what the pre trained language models are and how they work.

Researching selected pre trained language models, their strengths and weaknesses and comparing them.

Researching how to test selected pre trained language models.

Testing and comparing the selected pre trained language models.

BENG Software Engineering 04/03/2022 1

Additional Information / Knowledge Required:
Knowledge about pre trained language models.

Knowledge and understanding selected pre trained language models.

Knowledge and understanding about testing techniques of selected pre trained language models.

Information Sources that Provide a Context for the Project:
Terena Bell and Thor Olavsrud. (2020). What is natural language processing? The business benefits of NLP explained. CIO.

Terena Bell and Thor Olavsrud. (2021). What is NLP? Natural language processing explained. CIO.

Mozafari, M., Farahbakhsh, R., & Crespi, N. (2020). Hate speech detection and racial bias mitigation in social media based on BERT model. PloS One, 15(8), e0237861–e0237861. https://doi.org/10.1371/journal.pone.0237861

The Importance of the Project:
This project is very important because it may help to build better application allowing for communication between robot and human.

The Key Challenge(s) to be Overcome:
Understanding what the pre trained language models are and how they work.

Understanding the selected pre trained language models, their strengths and weaknesses.

Appendix B: Interim Report

SOC10101 Honours Project (40 Credits)

Week 9 Report

Student Name: JERZY BAJER

Supervisor: DIMITRA GKATZIA

Second Marker: KEHINDE OLUWATOYIN BABAAGBA

Date of Meeting: 29/11/2021

Can the student provide evidence of attending supervision meetings by means of project diary sheets or other equivalent mechanism? yes

If not, please comment on any reasons presented

Please comment on the progress made so far

The student has made progress and has completed the literature review phase on the project ahead of the time he scheduled to do this.

Is the progress satisfactory? yes

Can the student articulate their aims and objectives? yes

If yes then please comment on them, otherwise write down your suggestions.

The aims and objectives were well articulated by the student which is centred around the comparison of pretrained language models such as BERT, GPT-2 and their capabilities of showing common-sense.

* Please circle one answer; if no is circled then this must be amplified in the space provided

Does the student have a plan of work? yes

If yes then please comment on that plan otherwise write down your suggestions.

The student provides a clear plan of work and appears to be working ahead of time which is good.

Does the student know how they are going to evaluate their work? yes

If yes then please comment otherwise write down your suggestions.

The evaluation plan is clear as the student points out the fact that this would be done using similar style in related literature that compares the accuracy of the various models to be employed.

Any other recommendations as to the future direction of the project

The student's progress is satisfactory, and it would be great to see the work done by the student in comparing the various pretrained language models. I would advise that as the student progresses with their project that they do this in line with the learning outcomes for this module.

* Please circle one answer; if no is circled then this must be amplified in the space provided

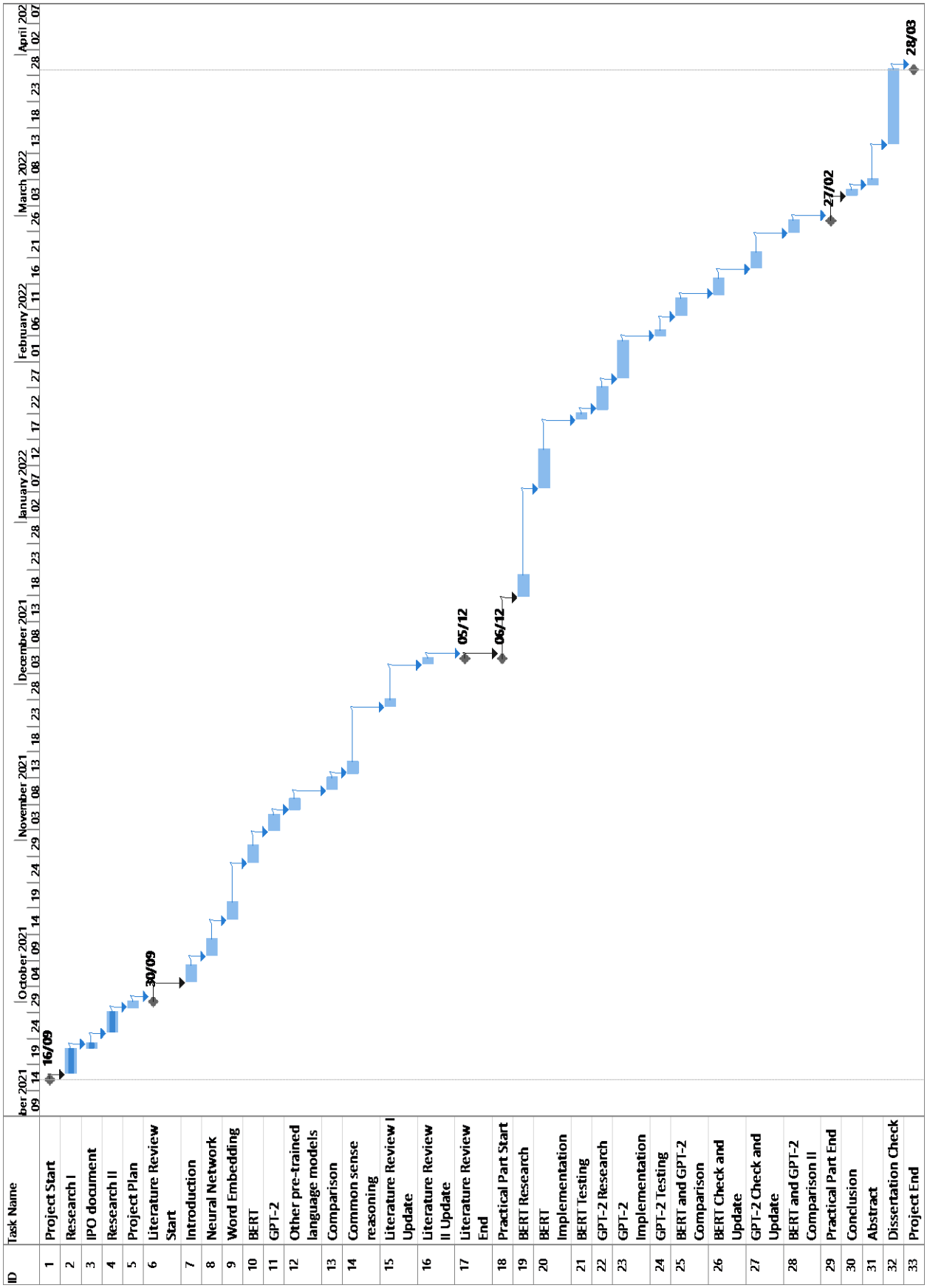
Signatures: Supervisor Second Marker
KEHINDEBABAAGBA

Student

The student should submit a copy of this form to Moodle immediately after the review meeting; A copy should also appear as an appendix in the final dissertation.

* Please circle one answer; if no is circled then this must be amplified in the space provided

Appendix C: Gantt Chart



Appendix D: Detailed Project Plan

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	Auto Scheduled	Project Start	0 hrs	Thu 16/09/21	Thu 16/09/21	
2	Auto Scheduled	Research I	20 hrs	Fri 17/09/21	Tue 21/09/21	1
3	Auto Scheduled	IPO document	4 hrs	Wed 22/09/21	Wed 22/09/21	2
4	Auto Scheduled	Research II	16 hrs	Sat 25/09/21	Tue 28/09/21	3
5	Auto Scheduled	Project Plan	8 hrs	Wed 29/09/21	Thu 30/09/21	4
6	Auto Scheduled	Literature Review Start	0 hrs	Thu 30/09/21	Thu 30/09/21	5
7	Auto Scheduled	Introduction	16 hrs	Mon 04/10/21	Thu 07/10/21	6
8	Auto Scheduled	Neural Network	16 hrs	Sat 09/10/21	Tue 12/10/21	7
9	Auto Scheduled	Word Embedding	16 hrs	Sat 16/10/21	Tue 19/10/21	8
10	Auto Scheduled	BERT	16 hrs	Wed 27/10/21	Sat 30/10/21	9
11	Auto Scheduled	GPT-2	16 hrs	Tue 02/11/21	Fri 05/11/21	10
12	Auto Scheduled	Other pre-trained language models	12 hrs	Sat 06/11/21	Mon 08/11/21	11
13	Auto Scheduled	Comparison	12 hrs	Wed 10/11/21	Fri 12/11/21	12
14	Auto Scheduled	Common sense reasoning	12 hrs	Sat 13/11/21	Mon 15/11/21	13
15	Auto Scheduled	Literature Review I Update	8 hrs	Fri 26/11/21	Sat 27/11/21	14
16	Auto Scheduled	Literature Review II Update	8 hrs	Sat 04/12/21	Sun 05/12/21	15
17	Auto Scheduled	Literature Review End	0 hrs	Sun 05/12/21	Sun 05/12/21	16
	Auto Scheduled	Practical Part Start	0 hrs	Mon 06/12/21	Mon 06/12/21	17
19	Auto Scheduled	BERT Research	20 hrs	Fri 17/12/21	Tue 21/12/21	18
20	Auto	BERT Implementation	32 hrs	Fri 07/01/22	Fri 14/01/22	19

	Scheduled					
21	Auto Scheduled	BERT Testing	8 hrs	Thu 20/01/22	Fri 21/01/22	20
22	Auto Scheduled	GPT-2 Research	20 hrs	Sat 22/01/22	Wed 26/01/22	21
23	Auto Scheduled	GPT-2 Implementation	32 hrs	Fri 28/01/22	Fri 04/02/22	22
24	Auto Scheduled	GPT-2 Testing	8 hrs	Sat 05/02/22	Sun 06/02/22	23
25	Auto Scheduled	BERT and GPT-2 Comparison	16 hrs	Wed 09/02/22	Sat 12/02/22	24
26	Auto Scheduled	BERT Check and Update	16 hrs	Sun 13/02/22	Wed 16/02/22	25
27	Auto Scheduled	GPT-2 Check and Update	16 hrs	Fri 18/02/22	Mon 21/02/22	26
28	Auto Scheduled	BERT and GPT-2 Comparison II	12 hrs	Fri 25/02/22	Sun 27/02/22	27
29	Auto Scheduled	Practical Part End	0 hrs	Sun 27/02/22	Sun 27/02/22	28
30	Auto Scheduled	Conclusion	8 hrs	Fri 04/03/22	Sat 05/03/22	29
31	Auto Scheduled	Abstract	8 hrs	Sun 06/03/22	Mon 07/03/22	30
32	Auto Scheduled	Dissertation Check	60 hrs	Mon 14/03/22	Mon 28/03/22	31
33	Auto Scheduled	Project End	0 hrs	Mon 28/03/22	Mon 28/03/22	32

Appendix E: Ethical Concerns

This project does not gather, store, or process any data that could be considered as sensitive. All data that is used to test the models' capabilities of showing common sense is general data without any sensitive content. Both alternative products dataset and kitchen utensils' description dataset are general datasets without any sensitive content. Therefore, there are no ethical concerns regarding this project. This project can be considered free of ethical concerns.

Appendix F: Project Diary

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date: 16.09.2021 **Last diary date:**

Objectives:

- Discussing the project.
- Getting more details regarding the project.
- Clarifying ambiguities.

Progress:

Dimitra described me the idea behind the project and the expectations regarding what is required to do. She answered my questions and clarified some ambiguities I had. She sent me some links and sked me to find out more regarding the pre-trained language models.

Supervisor's Comments:

Dimitra asked me to visit the links she provided to me and find out more about the pre-trained language models BERT and GPT-2.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date: 30.09.2021 **Last diary date: 16.09.2021**

Objectives:

- Further discussing the project.
- Gaining better understanding of the project.
- Clarifying ambiguities.

Progress:

I gained better understanding of the project and what is required. Dimitra clarified my ambiguities and answered the questions I had. I made the final decision to do this project.

Supervisor's Comments:

None

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:14.10.2021 **Last diary date: 30.09.2021**

Objectives:

- Discussing the IPO.
- Discussing the literature review plan.
- Discussing the progress.

Progress:

Dimitra gave me feedback regarding the IPO. She explained me what I must change. I was told that once I modify the IPO the way she told me I can upload it to the Moodle. She also provided me feedback regarding the literature review's plan. She explained me which parts I should add to the literature review and which parts I can remove from the literature review. I managed to clarify all ambiguities regarding the literature review.

Supervisor's Comments:

Dimitra asked me to add some sources to the IPO and to modify some parts and upload once ready.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:25.10.2021 **Last diary date: 14.10.2021**

Objectives:

- Discussing the Introduction part.
- Discussing the Neural Networks part.

Progress:

Dimitra provided me the feedback regarding the Introduction part. She asked me to add section regarding common sense to the Introduction. Dimitra also provided me the feedback regarding the Neural Network part of the literature review.

Supervisor's Comments:

Dimitra asked me to introduce the common sense in the Introduction section.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:4.11.2021 **Last diary date: 25.10.2021**

Objectives:

- Discussing the Word Embedding part.
- Discussing the project's progress.

Progress:

Dimitra provided me the feedback regarding the Word embedding part.

Supervisor's Comments:

Dimitra asked me to pay attention to some typo errors.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:11.11.2021 **Last diary date: 4.11.2021**

Objectives:

- Discussing the BERT part.
- Discussing the project's progress.

Progress:

Dimitra provided me feedback regarding the BERT part of the literature review.

Supervisor's Comments:

None

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:25.11.2021 **Last diary date: 11.11.2021**

Objectives:

- Discussing the GPT-2 part.
- Discussing the other parts.
- Discussing the comparison
- Discussing the 9th week meeting with the second marker.

Progress:

Dimitra provided me the feedback regarding the three parts I managed to complete and sent to her. She mentioned what to pay attention to and what eventually modify. I gained the knowledge what to expect during the 9th week meeting with the second marker.

Supervisor's Comments:

Dimitra told me that the 9th week meeting with the second marker will be a general discussion and that I will be asked some questions regarding my project.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:17.01.2022 **Last diary date: 25.11.2021**

Objectives:

- Discussing the practical part of the project.
- Discussing how to implement and test BERT model(s)
- Discussing how to implement and test the GPT-2 model(s)

Progress:

Dimitra explained me how to implement the BERT and GPT-2 models and how to use them to test their capabilities of showing common sense.

Supervisor's Comments:

Dimitra asked me to use both datasets to test the models' capabilities of showing common sense. She said I do not have to use all datasets, however, the more I use the better.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:24.01.2022 **Last diary date: 17.01.2022**

Objectives:

- Further discussing the BERT and GPT-2 models

Progress:

I gained better understanding how to test and compare the models. Using both the not fine-tuned and the fine-tuned models will provide additional information regarding the models' capabilities of showing common sense.

Supervisor's Comments:

Dimitra asked me to use both not fine-tuned and fine-tuned models so I can not only compare BERT with GPT-2 but also, I can compare the not fine-tuned models with the fine-tuned models.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:31.01.2022 **Last diary date: 24.01.2022**

Objectives:

- Discussing the testing of the BERT and GPT-2 models.
- Discussing the progress of the project

Progress:

I discussed the progress of the project with Dimitra. Dimitra provided me the feedback regarding the code I created.

Supervisor's Comments:

None

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:14.02.2022 **Last diary date: 31.01.2022**

Objectives:

- Discussing the progress of the project

Progress:

The practical part, the coding and the testing, started coming to an end.

Supervisor's Comments:

None

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:28.02.2022 **Last diary date: 14.02.2022**

Objectives:

- Discussing the Implementation part
- Discussing the Testing and Results part
- Discussing the Conclusion part
- Discussing the Abstract part
- Discussing the Code

Progress:

Dimitra provided me the feedback regarding all four parts I sent to her. I was asked to modify the parts and to add some parts as well. The report was getting to be ready soon.

Supervisor's Comments:

Dimitra asked me to change the real code in the Implementation part with the pseudocode. Dimitra asked me to add the table for the Testing and Results part and to create the separate Conclusion for the whole Project. Dimitra asked me also to add some examples for the Abstract.

PROJECT DIARY

Student: Jerzy Bajer **Supervisor: Dimitra Ghatzia**

Date:07.03.2022 **Last diary date: 28.02.2022**

Objectives:

- Discussing the Dissertation
- Discussing the Project

Progress:

I was asked to add additional examples to the testing, 10 for alternative products and 10 for kitchen utensils' description for each model. We agreed that once I put all parts together and add the Appendices I will send the whole dissertation to Dimitra to get the final feedback.

Supervisor's Comments:

Dimitra asked me to test all models for additional 10 examples for each dataset., She asked me to send her the whole dissertation once it gets ready to get the final feedback.