Primer-Selection Methods for Generating Reflections using GPT-2

Written by:

Eric Keilty

Supervisor: Jonathan Rose

April 2021

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Primer-Selection Methods for Generating Reflections using GPT-2

Written by: Eric Keilty

Bachelor's of Applied Science Thesis

Supervisor: Jonathan Rose

April 2021

Abstract

Few-shot learning is a powerful tool which allows GPT-2 the ability to learn tasks with very few examples. The method involves pre-pending examples, called primers, of the task correctly done to the beginning of the input. GPT-2 is then able to infer the correct output based on the pattern of the primers. One aspect that has yet to be explored in few-shot learning is how one should choose these primers. Three primer-selection methods are proposed and evaluated on the task of generating high-quality reflections. The first method is to choosing the primers randomly; this acts as the baseline method. The second is to choose the primers that are most similar in semantic meaning to the original input. The third method is to choose the primers that are most different, or least similar, in semantic meaning to the original input. Semantic similarity between a primer and the original input is measured using a universal sentence encoder, which creates a fixed-length embedding vector from arbitrary-length text, and ranking each primer using the cosine similarity measure. The primers with the highest similarity score to the input are considered the most "similar". The results of this research suggests that choosing semantically similar primers tends to produce more reliable and robust reflections compared to the other proposed methods. This shows that a primer-selection method exists that preforms better than selecting primers at random. This provides a stepping-stone for future research in creating more sophisticated primer-selection methods for improving the quality of any text-generation task.

Acknowledgements

Firstly, I would like to thank Prof. Jonathan Rose for the guidance and support he provided for the past one and a half years. Thank you for being an excellent mentor and always having my best interest at heart.

Additionally, I thank everyone in Prof. Rose's research group for facilitating a positive atmosphere in which to conduct this research. In particular, I thank Imtihan Ahmed for his work on the Reflection Quality Classifier, without which this thesis would not have been possible.

Thank you to Dr. Peter Selby and everyone at CAMH for their invaluably insight and feedback on the generation of reflections. Without their work in creating high-quality reflections for our dataset, this thesis would not have been possible.

I thank my friends and family for their continued encouragement and support. I would not be where I am today without them.

Finally, I would like to thank Prof. Alan Chong, Prof. Lisa Romkey, and the Department of Engineering Science for their support in the undergraduate thesis program, and allowing this research to be possible.

Contents

List of Figures

List of Tables

1 Introduction

Motivational Interviewing (MI) is a well-established client-centered style of counseling designed to increase a client's motivation to change their behavior by addressing ambivalence [1]. Originally developed as a treatment for alcohol addiction, MI has been recently found to yield "a modest but significant increase" in helping smokers successfully quit [2]. Furthermore, computer-based conversational system (chatbot) implementations of MI-style conversations have also been shown to have a positive impact on smokers [3]. However, current chatbot implementations are limited by the predetermined structure of the conversation. Much of the negative feedback of these systems is related to the repetitive nature of the questions or the system misunderstanding the client [3]. With only 60 percent of smokers ever quitting and less than one-third reported to use counseling when trying to quit [4], creating more robust chatbots that are both more accessible and more capable of having better MI-style will help increase smoking cessation.

One of the key skills needed to have an MI-style conversation is the ability to create high-quality reflections. In the context of MI, a *reflection* is a statement intended to mirror the meaning of preceding client speech. In particular, there are two types of reflections. A *simple reflection* contains little or no additional content, and is similar to a restatement. Conversely, a *complex reflection* infers additional meaning beyond what the client has said [1]. Any MI-style chatbot needs to be able to generate high-quality reflections, both simple and complex. Thus, the broad focus of this research is centered around methods for creating high-quality reflections using a fully generative model.

In recent years, the transformer architecture has dominated the NLP space. The original transformer architecture involved stacked encoders and decoders, which the utilized attention mechanism in order to model sentence dependencies [5]. GPT-2, which consists only of decoder stacks, is currently the best performing pre-trained, text-generation, transformer model publicly available. Therefore, this will be the primary model used to generate reflections. Traditionally for a pre-trained model to perform a specific task, one would need to re-train it using many correct examples; this process is called *fine-tuning*. However, the development of GPT-2's successor (GPT-3) introduced a new method called *few-shot learning*, which involves pre-pending examples of the task correctly done to the beginning of the input. Thus, the new input to the model consists of a number of pre-pended examples concatenated with the original input. GPT-2 is then able to infer the correct output based on the pattern of the pre-pended examples [6]. The main benefit of few-shot learning is it requires significantly less training data. With only a handful of examples, few-shot learning can achieve similar performance to a fine-tuned model re-trained on 1,000-10,000 examples [6]. This is preferred in tasks where obtaining correct examples is difficult, as is the case for this problem.

One aspect that has yet to be explored in few-shot learning, and the main focus of this research, is how one should choose the pre-pended examples, which will be referred to as *primers*. The easiest approach is to simply choose a static set of primers that are used for every input. Alternatively, I propose a dynamic approach where the primer set is chosen at run-time from a bank of possible examples based on the given input. In particular, I propose three primer-selection methods. First, choose the primers randomly; this will act as the baseline. Second, choose the primers that are the most "similar" to the given input. Finally, choose the primers that are the least "similar" or most "different" to the given input. The similarity of a primer to a given input will be determined by using a *universal sentence encoder*, which creates a fixed-length embedding vector from arbitrary-length text, and ranking each primer using the *cosine similarity* measure. The primers with the highest similarity score to the input are considered the most "similar".

The goal of this thesis is to evaluate these proposed primer-selection methods to determine which method produces the most reliable and highest quality reflections. Thus, this research aims provide a stepping-stone to future research in creating other methods for improving the quality of any text generation task.

2 Background

This thesis consists of the application of generative models to the practice of Motivational Interviewing targeting smoking cessation. Thus, the background will begin with a brief overview of smoking cessation and Motivational Interviewing. Then, it will continue with an overview of language models. This will aid in understanding the general transformer architecture as originally proposed, and subsequently the GPT-2 architecture.

2.1 Motivational Interviewing Applied to Smoking Cessation

2.1.1 Motivational Interviewing

Motivational Interviewing (MI) is a client-centered counseling style designed to address ambivalence and motivate positive behavior change in clients. Rather than being a specific technique, MI provides a framework under which to operate, referred to as the *spirit of MI*. In MI, the practitioner and the client engage in an active collaboration built on mutual understanding. Nothing is forced or imposed onto the client, rather the role of the practitioner is to evoke the wisdom and the strengths that the client already contains within themselves. This is accomplished through the use of four core skills: *open questions*, *affirmations*, *reflections*, and *summaries*. An open question is a question that does not have a predictable answer and leaves room for the client to elaborate. An

affirmation is an acknowledgment of a person's positive effort. A reflection is a statement intended to mirror what has been previously said. Finally, a summary is a concise restatement of one or more prior client statements. Open questions and affirmations emphasize the client-centered nature of practicing MI. Reflections and summaries allow for mutual understanding between the client and the practitioner [1].

This thesis is particularly focused on reflections, of which there are two types. A *simple reflection* contains little or no additional content, and is similar to a restatement. Conversely, a *complex reflection* infers additional meaning beyond what the client has said. In order to create a good refection, one needs to demonstrate empathy and a deep understanding of what the client is saying [1]. An example and comparison of a simple and complex reflection can be found in Appendix A.

2.1.2 Smoking Cessation

Smoking cessation is the sustained discontinuation of smoking cigarettes or use of other tobacco products [7]. Since smoking cessation for any amount of time has been shown to reduce the negative effects of smoking while increasing life-expectancy [8], research in this area has been focused on intervention strategies to encourage patients to quit. Traditional interventions include Nicotine Replacement Therapy in the form of nicotine gum or nicotine patches, and clinical methods in the form of personal counseling or support groups [7].

Unfortunately, these methods have not proved to be sufficient. Less than 10 percent of smokers are considering quitting in the immediate future [9], and only about 40 percent of all smokers make an attempt to quit each year [10]. Therefore, in recent years Motivation Interviewing has been applied to smoking cessation. By targeting the ambivalence towards quitting, MI has been found to yield "a modest but significant increase" in helping smokers successfully quit [2]. However, less than one-third of all smokers reported to use counseling when trying to quit [4]. Thus, there is a need for a solution that can provide the benefits of MI to those who do not seek counseling.

2.1.3 Chatbot Implementations of Motivational Interviewing

As part of his Master's Thesis, Fahad Almusharraf created one of the first computer-based conversational system (chatbot) implementations of MI. This system utilizes the *running head start* method, which begins the discussion by asking questions to evoke the pros and cons of behavior change. The conversation was modeled as a finite state machine (FSM) where each node represented a type of response the chatbot could give. For example, a node could be associated with asking an open question, summarizing/reflecting on what was previously said, or asking for clarification from the client. With a small amount of training, Fahad's chatbot could interpret the intent

of the client's text and determine the most appropriate next node in the FSM. The chatbot would then randomly output one of several pre-written responses associated with that node. The goal of the conversation design was to elicit self-reflection in the client through a series of pre-written open questions and summarizations [3].

In a study of 221 subjects, Fahad's MI chatbot proved to have a positive impact on smokers. However, much of the reported frustration from the participants was related to the limitation of the chatbot's utterances. Since the chatbot's statements were pre-written, if the subject's response didn't fit into the template of the conversation then the chatbot was unable to provide an adequate response. This caused frustration among the participants and diminished the effectiveness of the chatbot [3]. Thus, there is room to improve this chatbot by creating a more robust system, capable of generating better and more diverse responses. In particular, rather than selecting from a bank of pre-written responses, the chatbot should be able to generate responses dynamically at runtime tailored towards what was previously said. This requires a model that has a more complex understanding of the English language.

2.2 Language Models

The study of language models is a vast field. Thus, the scope of this background will be limited to only what is necessary to understand the transformer architecture, as this is the basis of the GPT-2 architecture.

2.2.1 Statistical Language Models

Almost every language model is a *statistical language model*, which is a probability distribution over all sequences of words. Let $P(\boldsymbol{s})$ be the true probability distribution of every sequence $\boldsymbol{s}$ in the given language, i.e. $P(\boldsymbol{s})$ represents the probability that the sequence $\boldsymbol{s}$ appears in any text written in the given language. The goal of all statistical language models is to sufficiently estimate $P(\boldsymbol{s})$. This is accomplished by analyzing a large body of text, called a *corpus*, and using a statistical algorithm in order to assign a probabilistic value to each sequence of words. Almost all language models decompose sequences of words into a product of conditional probabilities using the chain rule of probabilities [11, 12].

$$P(\boldsymbol{s}) = P(w_1,\ \ldots,\ w_L) = \prod_{i=1}^{L} P(w_i | w_1,\ \ldots,\ w_{i-1}) \tag{1}$$

Classically, the next step of the statistical algorithm consists of various forms of statistical estimation in conjunction with smoothing techniques. More recently, this statistical estimation is

accomplished using deep neural networks such as GPT-2. This is discussed in Section 2.4.1.

2.2.2 Text Generation with Auto-Regressive Models

Almost every modern text generation model is an *auto-regressive model*, which assumes the output depends on a combination of past values. Mathematically this is written as $P(w_{L+1}|w_1, \ldots, w_L)$, i.e. the next word the model generates depends on all previous words. This means text generation models are recursive in nature. The word that was generated will become part of the input to generate the next word [13]. This recursive scheme is shown in Figure 1. Notice, this architecture allows for arbitrary-length output.

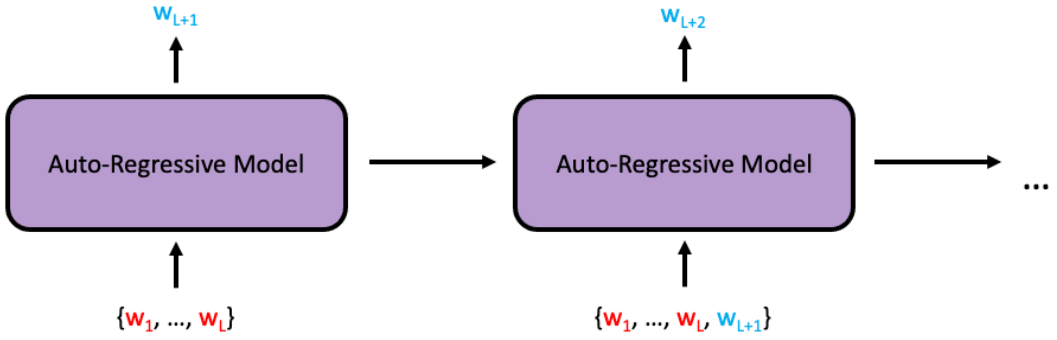

Figure 1: Text Generation using an Auto-Regressive Model

2.2.3 N-Grams

In an ideal world, these auto-regressive language models would use the full history of the text ($\{w_1, \ldots, w_L\}$) when generating the next word (w_{L+1}). However, in practice we are limited by the computational power available. The length of the sequence, L, could be hundreds if not thousands of words long. Instead, we restrict the model to only consider the previous $n-1$ words. This sequence of n contiguous words is called an *n-gram*, and is sometimes referred to as the *context window* of the model. In Equation 1, this translates to the Markov Assumption of order $n-1$, i.e. that each word in the sequence only depends on the previous $n-1$ words [12, 13].

$$P(w_i|w_1, \ldots, w_{i-1}) \approx P(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) \tag{2}$$

Thus, Equation 1 becomes the following [11, 13].

$$P(\boldsymbol{s}) = P(w_1, \ldots, w_L) \approx \prod_{i=1}^{L} P(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) \tag{3}$$

Finally, Figure 1 becomes the following.

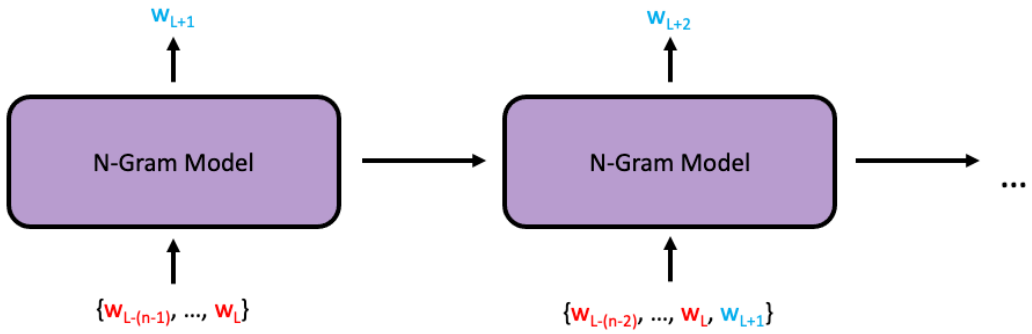

Figure 2: Text Generation using an N-Gram Model

2.2.4 Word Vectors

The last major piece of modern language models is how words are represented. Traditionally, words were represented as categorical data, which is called the *Bag of Words* model. The language model would then have an input node for each word in the vocabulary, and one could use boolean word occurrence, word frequency, or the TF-IDF score as the input value to each node.

In 2013, Tomas Mikolov and a team of researches at Google created the *Word2Vec* algorithm, which allowed for words to be represented as continuous vectors that live inside a latent vector space [14]. The architecture for achieving this can be found in Appendix B. These vector representation of words are called *word vectors* or *word embeddings*. The main advantage of embedding words as continuous vectors is it allows information to be shared between related words. In particular, words used in similar contexts will be close together in this latent vector space. This is called a *Distributed Representation* of words, since the information about the word is represented collectively by the entire vector space, rather than locally in a single variable [13].

Every modern language model either uses pre-trained word vectors or learns its own word vectors implicitly. GPT-2 does the latter with some slight additions. This is discussed in Section 2.4.1.

2.3 The Transformer Architecture

The original transformer architecture involved stacked encoders and decoders, which utilize the attention mechanism in order to model sentence dependencies [5]. Thus, the Transformer architecture can be decomposed into 3 key elements: the encoder/decoder architecture, the attention mechanism, and feed-forward networks. Figure 3 shows the full transformer architecture as originally proposed [5].

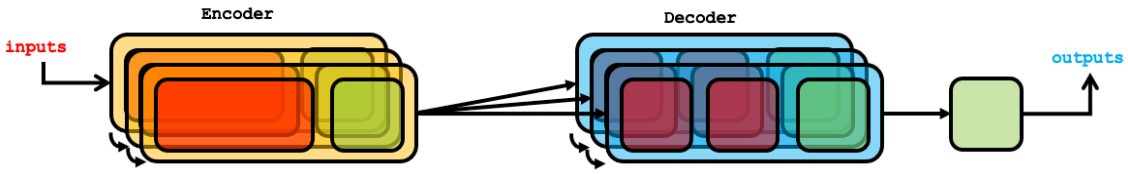

Figure 3: Simplified "Attention is All You Need" Transformer Architecture

Encoder blocks are displayed in yellow, decoder blocks are displayed in blue, attention networks are displayed in red, and feed-forward networks are displayed in green. A feed-forward network is simply a series of fully-connected or linear layers. Thus all that remains is to discuss the encoder/decoder blocks and the attention networks.

2.3.1 The Encoder/Decoder Architecture

The idea of the encoder/decoder architecture is to split the job of the model into 2 subtasks. The encoder's task is to condense the raw input sequence into a vector representation. This will be refered to as the *encoder output* or the *code vector*. The job of the decoder is to then generate a new sequence auto-regressively based on the encoder output and what the decoder previously generated. The high-level architecture is shown in Figure 4.

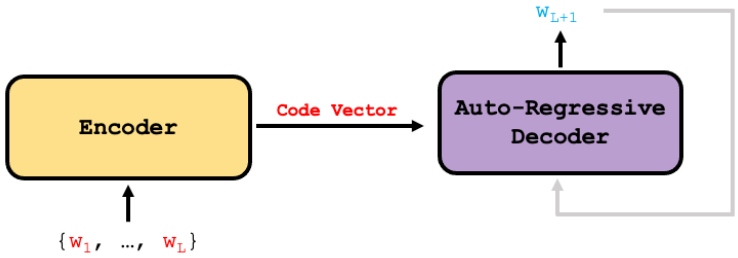

Figure 4: The High-Level Encoder/Decoder Architecture

In Figure 3, there are multiple encoder blocks that make up the entire encoder of the transformer, and likewise for the transformer decoder. The output of the previous block becomes the input to the next block. This in another important feature of the transformer. Stacking encoder/decoder blocks allows us to add parameters to the model without bound. In practice when transformers are applied to general language models, it has been found that adding more encoder/decoder blocks generally improves the quality of the model output [6, 15]. Thus, the transformer architecture is easily scaleable as the level of compute increases. Moreover, splitting the computation into several blocks allows this architecture to be highly-parallelizable on GPUs [5].

2.3.2 The Attention Mechanism

The attention mechanism allows the transformer to model long-distance sentence dependencies. In Section 2.2.3, it was discussed that many language models utilize n-grams in order to make training their model tractable. In practice, models often cannot use n-grams larger than 7 due to computational restrictions. This posed a problem for text generation models, as the model was unaware of anything generated outside of this relatively small context window. Attention is an efficient way for models to increase the size of their context window and often allows for the entire history to be taken into account.

The attention mechanism consists of an attention network that takes in a word and a sequence as input, and outputs a weight often called the *context*. The exact architecture of this attention network is given in Appendix C. In its most simplified form, it can be written as the following function [5].

$$\texttt{Attention}(w, \boldsymbol{s}) = c \tag{4}$$

This attention weight, c, is intuitively interpreted as how important word w is when deciding the next state of sequence $\boldsymbol{s}$. More mathematically precise, it is a minimum entropy projection onto the space of probability measures [16].

In general, word w could be any word and sequence $\boldsymbol{s}$ could be any sequence. *Self-attention* is when word w is part of sequence $\boldsymbol{s}$ [5]. Thus, Equation 4 is simplified to the following.

$$\texttt{Self-Attention}(\boldsymbol{s})_i = \texttt{Attention}(s_i, \boldsymbol{s}) \tag{5}$$

Finally, *masked self-attention* is when we only consider previous values of the sequence. This is utilized in text generating transformers as they are auto-regressive. Thus, during training, they should not have access to future words in a sentence.

$$\texttt{Masked-Self-Attention}(\boldsymbol{s})_i = \texttt{Attention}(s_i, \{s_1, \ldots, s_i\}) \tag{6}$$

The reason attention is so powerful is because it is fast to compute and highly parallelizable. This allows for the context window of the transformer model to be hundreds of words long, which dramatically improves the coherence of model's generated text.

2.4 GPT-2

2.4.1 The GPT-2 Architecture

GPT-2 is the decoder stack of the transformer architecture in Figure 3. A breakdown of the architecture is shown in Figure 5.

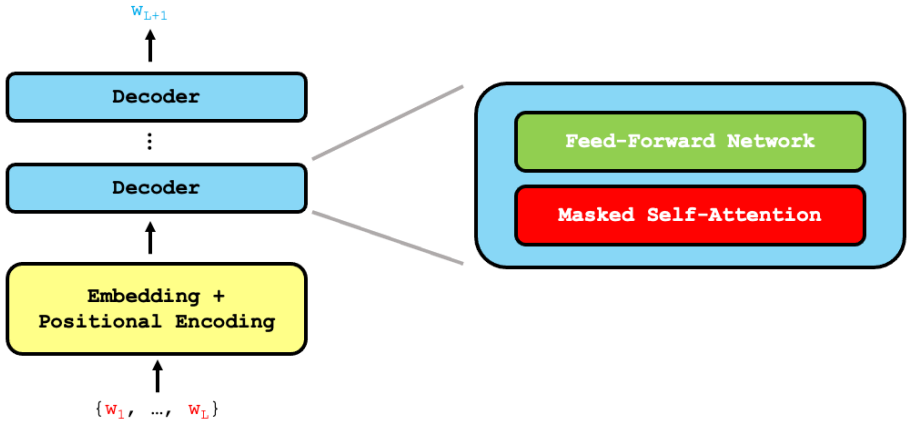

Figure 5: The GP2 Architecture

The input to GPT-2 is simply a sequence of words. In its training, GPT-2 has learned its own word vectors. This is stored in the embedding matrix of size (`vocab_size` $\times$ `embedding_size`), where each row corresponds to a word vector. GPT-2 tokenizes the input sequence to create a matrix of size (`input_size` $\times$ `embedding_size`). Positional encodings are added to the corresponding positions in this new matrix, which tell the decoder blocks where each word appears in the input sequence. Masked self-attention is then applied to this matrix along with a feed-forward network, which arrives at a matrix of size (`input_size` $\times$ `embedding_size`), which can then be fed into the next decoder block [15, 17].

At the final decoder block, GPT-2 outputs a matrix of size (`input_size` $\times$ `embedding_size`). However, we only care about the last row of this matrix, which corresponds to the last word of the input sequence. This vector is multiplied by the embedding matrix, which produces a vector of size (`vocab_size` $\times$ 1) containing logits corresponding to each word in the vocabulary. Recalling Section 2.2.2, this is the estimation of the probability distribution $P(w_{L+1}|w_1, \ldots, w_L)$ [15, 17].

One last thing to note is that GPT-2 is an auto-regressive model. Exactly as shown in Figure 1, once GPT-2 generates w_{L+1}, it adds this generated word to the input, and preforms the entire process again in order to generate w_{L+2}.

2.4.2 GPT-2 Decoding Methods

A decoding method refers to selecting word w_{L+1} from the distribution $P(w_{L+1}|w_1, \ldots, w_L)$. There are three decoding methods GPT-2 has implemented: greedy, beam search, and sampling.

The *greedy decoding method* selects the highest probability next word at each step [18].

$$w_{L+1} = \underset{w}{\text{argmax}} \; P(w_{L+1}|w_1, \ldots, w_L) \tag{7}$$

The *beam search decoding method* is similar to greedy, except it takes the top `beam_num` highest-probability words at each step. Figure 6 shows an example for `beam_num` $= 2$ [18].

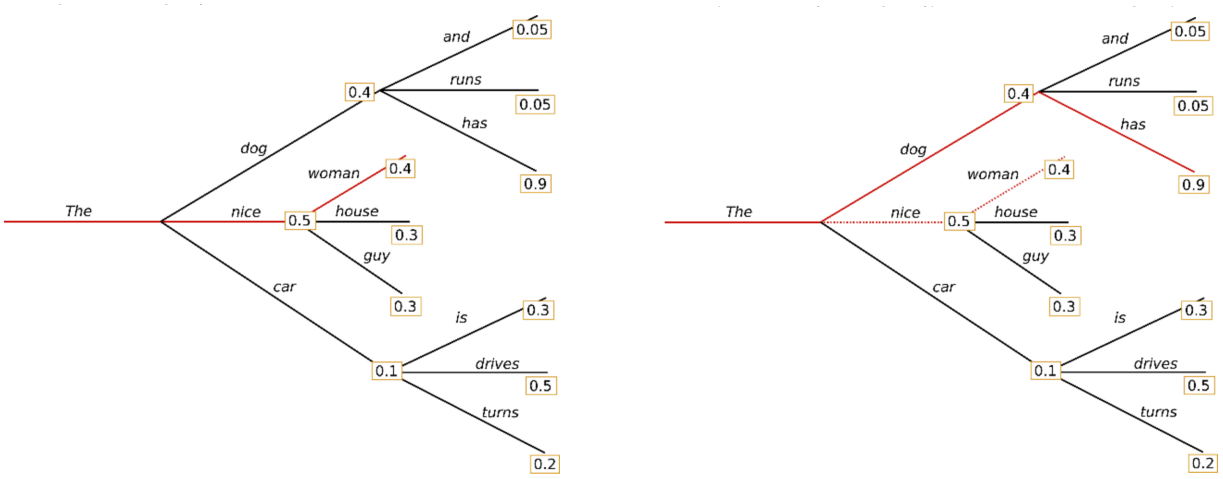

Figure 6: Comparison of Greedy (left) and Beam Search (right) Decoding Methods

Finally, the *sampling decoding method* randomly samples from this distribution [18].

$$w_{L+1} \sim P(w_{L+1}|w_1, \ldots, w_L) \tag{8}$$

One problem with pure random sampling is that it can result in incoherent outputs. There are three hyperparameters that have been created in order to improve this: Temperature, Top K, and Top P.

Temperature makes the distribution sharper, i.e. increases the likelihood of high-probability words and decreases the likelihood of low-probability words. As Temperature approaches 0, the sampling method will approach the greedy method [18].

Top K truncates the distribution to the top k words, and then redistributes the probability mass among the remaining words. As k approaches 1, the sampling method will approach the greedy method [18, 19]. Note that when using GPT-2 setting `top_k` $= 0$ is the same conceptually as setting

k to infinity, meaning the distribution will not be truncated.

Top P chooses the smallest set of words whose cumulative probability exceeds the probability p. The probability mass is then redistributed among the remaining words. As p approaches 0, the sampling method will approach the greedy method [18, 20].

2.4.3 Few-Shot Learning

Traditionally for a pre-trained model to perform a specific task, one would need to re-train it using many correct examples; this process is called *fine-tuning*. However, the development of GPT-2's successor (GPT-3) introduced a new method called *few-shot learning*, which involves pre-pending examples of the task correctly done to the beginning of the input. These pre-pended examples will be referred to as *primers*. Thus, the new input to the model consists of a number of primers concatenated with the original input, or the *query*. GPT-2 is then able to infer the output of the query based on the pattern of the primers [6]. An example of this is show in Figure 7. Note that a task description is not pre-pended to the input in this research application.

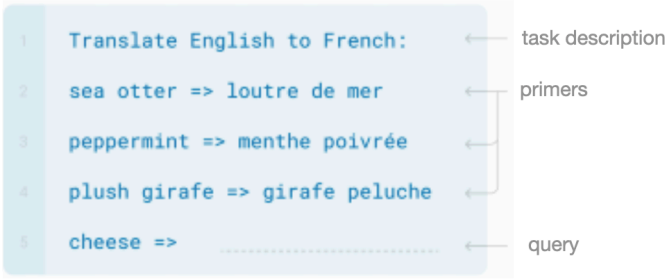

Figure 7: GPT-2 Few-Shot Example

The main benefit of few-shot learning is it requires significantly less training data. With only a handful of examples, few-shot learning can achieve similar performance to a fine-tuned model re-trained on 1,000-10,000 examples [6]. This is preferred in tasks where obtaining correct examples is difficult, which is the case for the task of generating high-quality reflections.

Surprisingly, what is not discussed in the literature is how one should choose these primers. Recently, there has been some research discussing the nature of the primers, but there is currently no research that recommends a method for selecting them. This is the topic of research for this thesis.

3 Primer Selection Methods

3.1 Dynamic Primer Selection

The naive way of selecting primers is to use the same set of primers for every query. This will be referred to as a *static primer-selection method*. In this scheme, a single set of high-quality primers are either selected or created, and are used in the GPT-2 input regardless of the query. The pseudo-code for this method is shown below.

```
STATIC-PRIMER-SELECTION(Query)
1: PrimerSet = P_1 || P_2 || ... || P_n
2: GPT2_input = PrimerSet || Query
3: return GPT2_input
```

The fundamental issue with this approach is that the same primer set may not be optimal for every query. GPT-2's output is highly dependent on the primers selected [21]. This begs the question of whether there is a more intelligent approach to choosing the primer set.

Therefore, I am proposing a *dynamic primer-selection method* where the primers are selected from a larger primer dataset at run-time based on the given query. The pseudo-code for this method is shown below.

```
DYNAMIC-PRIMER-SELECTION(Query, PrimerDataset)
1: PrimerSet = PRIMER-SELECTION-METHOD(Query, PrimerDataset)
2: GPT2_input = PrimerSet || Query
3: return GPT2_input
```

The job of the `PRIMER-SELECTION-METHOD` function is to choose primers that will optimize GPT-2's performance on a given query. The goal of this thesis is to show that such a function exists that can preform better than the static primer-selection method.

3.2 Universal Sentence Encoders

In order to construct the `PRIMER-SELECTION-METHOD` function, we need a method of comparing each primer in the primer dataset to the given query. The main obstacle is that a primer and a given query could be any string of any length. Using the raw text, it would be very difficult to develop an algorithm that is able to compare any primer to any query.

Therefore, I propose using a *universal sentence encoder*, which is a pre-trained neural network model that creates a fixed-length embedding vector from arbitrary-length text. Thus, each primer

and the given query will be mapped to vectors of the same length, allowing them to be easily compared. More importantly, these fixed-length vectors live in a latent vector space, meaning vectors that are closer together in this space are similar in semantic meaning. This is shown in Figure 8.

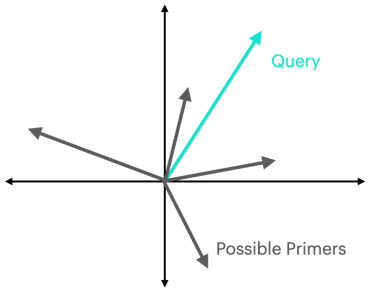

Figure 8: Latent Vector Representation of the Primer Dataset and a Given Query

Having this latent vector representation is very powerful, and is critical to creating the various `PRIMER-SELECTION-METHOD` functions.

3.3 Proposed Primer-Selection Methods

The goal of this thesis is to show a `PRIMER-SELECTION-METHOD` function exists that can preform better than the naive static primer-selection method. I propose the following dynamic primer-selection methods.

- *"random" primer-selection method*: selects n primers randomly from the primer dataset. This will act as the baseline method. Since the primers selected have no correlation to the given query, this method is analogous to a static primer-selection method.
- *"similar" primer-selection method*: selects the top n primers that are most "similar" to the given query.
- *"different" primer-selection method*: selects the top n primers that are most "different" or the least "similar" to the given query.

The proposed methods are based on the concept of "similarity" between the primers and the given query. The similarity of a primer to a given query will be determined by the universal sentence encoder by ranking each primer using the *cosine similarity* measure. The primers with the highest similarity score to the input are considered the most "similar". This is taking advantage of the fact that the vectors live in a latent vector space. Therefore, vectors closer together in the space, have a more similar semantic meaning. It is also important to note that we are using cosine similarity rather than Euclidean similarity. This is because empirically it is generally accepted that direction is

much more important than magnitude for obtaining semantic similarity. This is explained in more detail in Appendix D. Below is pseudo-code for each proposed method.

`RANDOM-PRIMER-SELECTION-METHOD`(Query, PrimerDataset)

Let n denote the number of primers the algorithm will select
PrimerSet = n random primers from PrimerDataset
GPT2_input = PrimerSet || Query
return GPT2_input

`SIMILAR-PRIMER-SELECTION-METHOD`(Query, PrimerDataset)

Let n denote the number of primers the algorithm will select
QueryVector = `USE`(Query)
PrimerVectors = `USE`(PrimerDataset)
SimilarityScores = `COSINE-SIMILARITY`(QueryVector, PrimerVectors)
Sort SimilarityScores
PrimerSet = the n primers in PrimerDataset with the **highest** values in SimilarityScores
GPT2_input = PrimerSet || Query
return GPT2_input

`DIFFERENT-PRIMER-SELECTION-METHOD`(Query, PrimerDataset)

Let n denote the number of primers the algorithm will select
QueryVector = `USE`(Query)
PrimerVectors = `USE`(PrimerDataset)
SimilarityScores = `COSINE-SIMILARITY`(QueryVector, PrimerVectors)
Sort SimilarityScores
PrimerSet = the n primers in PrimerDataset with the **lowest** values in SimilarityScores
GPT2_input = PrimerSet || Query
return GPT2_input

Figure 9 shows an example of these three methods where the primer dataset consists of four primers, and the primer-selection method is tasked with picking $n = 2$ primers.

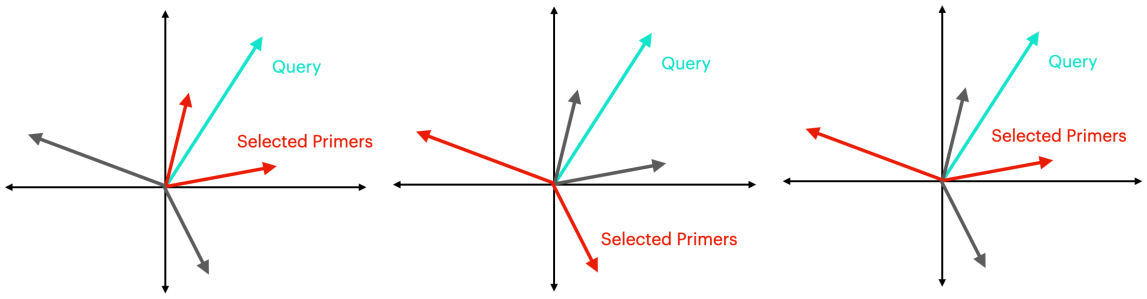

Figure 9: Similar (left), Different (center), and Random (right) Primer-Selection Methods

4 Evaluation Methods

4.1 The Primer Dataset

The dataset that is used to evaluate the proposed primer-selection methods comes from the conversations of Fahad's chatbot discussed in Section 2.1.3. The conversation transcripts were parsed for each prompt/response pair, where a *prompt* is an open question asked by the chatbot and a *response* is the human participant's response to the prompt. Additionally, each prompt/response pair has a corresponding reflection. These reflections have either been written or validated by practitioners at Centre for Addiction and Mental Health (CAMH) with each reflection being given a score by CAMH corresponding to its quality. This allowed the reflections to be split between simple and complex. Thus, there are two datasets: one where each row contains a prompt/response/simple-reflection triple, and the other where each row contains a prompt/response/complex-reflection triple. These datasets will be referred to as the *simple-reflection dataset* and the *complex-reflection dataset* respectively.

Additionally, each row of the dataset is given two more labels: Type and Topic. Since the questions asked by Fahad's chatbot (the prompt) were hard-coded, many of them are identical with the exception of a few words. Moreover, the words that would be inserted into these templates came from a hard-coded list of topics, such as smoking routine, health, cost, etc. Thus, *Type* refers to the classification of the various types of questions the chatbot asked, and *Topic* refers to the classification of the various topics of the questions. The reason for this classification is discussed in Section 4.4 and 4.5. Table 1 summarizes the fields in the simple and complex primer datasets.

Prompt	**Response**	**Reflection**	**Type**	**Topic**
Open Question asked by Fahad's Chatbot	Human participant's response to the prompt	CAMH created/verified reflection to the prompt/response pair	The type of the question being asked	The topic of the question being asked

Table 1: Description of the Fields in the Primer Dataset

4.2 The Reflection Quality Classifier

In order to analyze the quality of the reflections being produced by GPT-2, we need a method of determining their quality. As part of his current Master's research, Imtihan Ahmed created the *Reflection Quality Classifier*, or the RQC. This is a fine-tuned BERT model trained on the boolean classification task of whether a reflection is valid for the given prompt/response pair. The RQC is reported to have an sensitivity of around 80 percent and an accuracy of around 75 percent.

The details of how it was fine-tuned is out of the scope of this thesis. For the purposes of this research, we will treat it as a black-box model that can determine the quality of a reflection to a prompt/response pair.

4.3 Hyperparameter Search

Before evaluating the proposed primer-selection methods, we need to establish what hyperparameters of GPT-2 will be used. This will be done using a coarse grid search.

4.3.1 Grid Search Hyperparameters

Recall from Section 2.4.2 there are three different decoding methods for GPT-2: greedy, beam search, and sampling. There are no additional hyperparameters for the greedy decoding method. Beam search is being omitted from this analysis for two reasons. First, it is drastically slower than both greedy and sampling. Second, it does not produce reflections that are sufficiently different than the greedy decoding method. The last decoding method, sampling, has 3 possible hyperparameters that could be altered: temperature, top k, and top p. The meanings of these hyperparameters are outlined in Section 2.4.2.

A very coarse grid search is preformed on the greedy decoding method and the hyperparameters of the sampling decoding method. Since both top k and top p truncate the distribution, it is slightly redundant to alter both values in a grid search. Therefore, to save computational resources, top p is fixed at the value of 1.0 and only temperature and top k are altered. The hyperparameters for the sampling decoding method are then split into three categories: low variance, medium variance, and high variance. The precise hyperparameter values in each category are given in Table 2.

	temperature	**top k**
Greedy	N/A	1
Low Variance	0.1	10
Medium Variance	0.5	50
High Variance	1.0	100

Table 2: Meanings of Hyperparameter Classification

Recall from Section 2.4.2 that the sampling decoding method randomly samples from the distribution $P(w_{L+1}|w_1, \ldots, w_L)$. Since top k truncates this distribution and temperature sharpens this distribution, a lower value of each will bias the sampling towards the most probable words in the distribution. This is the intuition behind the category names: low, medium, and high variance. The smaller the values of temperature and top k, the more deterministic the sampling and the closer it becomes to the greedy decoding method. The greedy decoding method can be thought of as a "Zero Variance" category as it always gives the same output for the same input.

4.3.2 Permutation of the Primer Set

The output of GPT-2 is highly volatile to the order of the primers in the primer set [21]. For example, using PrimerSet $= P_1 \mid\mid P_2 \mid\mid P_3$ may produce a vastly different output compared to PrimerSet $= P_3 \mid\mid P_2 \mid\mid P_1$. Therefore, for each query, a number of different permutations of the chosen primers will be used to generate multiple reflections with GPT-2. This allows the variance caused by the order of the primers to be accounted for in the hyperparameter search.

4.3.3 Grid Search Procedure

Ideally, there would be separate query datasets and primer datasets, where the query dataset contains prompt/response pairs and the primer dataset contains prompt/response/reflection triples. However, since the available data is limited and this is preliminary analysis, the query dataset and the primer datasets are the same.

The grid search begins by iterating over each query (prompt/response pair) in the query dataset. The primer set is then selected randomly from the all primers in the primer dataset excluding the one containing the current query. The random primer-selection method is used rather than the similar or different primer-selection method since random is the baseline method. The selected primers are then randomly permuted a number of times to produce multiple reflections. Finally, each hyperparameter category is iterated over, including greedy, low-variance, medium-variance, and high-variance. A reflection is generated by GPT-2 and labeled by the RQC. The full grid search procedure is outlined in the pseudo-code below. Note that this procedure is preformed on both the simple-reflection primer dataset and the complex-reflection primer dataset.

```
GRID-SEARCH(PrimerDataset)
 1: QueryDataset = PrimerDataset
 2: for each Query in QueryDataset do
 3:     remove Query from PrimerDataset
 4:     PrimerSet = RANDOM-SELECTION-METHOD(Query, PrimerDataset)
 5:     for p = 1 to number of permutations do
 6:         permute PrimerSet
 7:         for hp in hyperparameter categories do
 8:             GPT2_input = PrimerSet || Query
 9:             reflection = GPT2(GPT2_input, hp)
10:             label = RQC(Query, reflection)
11:             append reflection and label to the relevant data-structures
12:         end for
13:     end for
14: end for
```

4.4 Leave-One-Group-Out Cross Validation

There are many possible definitions of the "best primer-selection method". One definition is the primer-selection method that best generalizes to unseen data. In other words, what primer-selection method (similar, different, or random) most reliably produces good reflections on queries not contained in the primer dataset?

4.4.1 Measuring Generalizability on a Small Dataset

Leave-one-out cross validation is a popular method for measuring the generalizability of models when one has a small dataset. The procedure consists of training the model over all but one row of the dataset, and then testing the model on the row that was "left out". In the case of few-shot learning for reflection generation, "training" is analogous to pre-pending the primers to the query, and "testing" is analogous to generating a reflection and labeling using the RQC. This process is repeated for each row of the dataset, and the result is a set of predictions by the model for each row in the dataset. This procedure measures generalizability because each time the model makes a testing prediction it was not trained on that data. In our case, the "prediction" is the generated reflection and associated label.

The leave-one-out cross validation could be applied to testing the primer datasets by removing one row from the primer dataset to be used as the query, and choosing the primers from the remaining rows as the primer dataset. However, the problem is the fact that many of the rows in Fahad's dataset are very similar. This is because Fahad's chatbot was hard-coded, and therefore as discussed in Section 4.1 each prompt comes from a fixed list of question templates and topics. Thus, some rows in this dataset are nearly identical.

4.4.2 The Procedure

I propose a slight modification to the traditional leave-one-out cross validation method, which is called *leave-one-group-out cross validation*. Recall the additional classifications (Type and Topic) of each row of the primer dataset in Section 4.1. The primer dataset can now be grouped by either Type or Topic. For example, if we are testing the Topic grouping, the "Groups" might be the list ['Health', 'Cost', 'Stress']. Instead of only removing one query from the primer dataset, this procedure removes the entire group from the primer dataset. This eliminates much of the self-similarity in Fahad's dataset, and therefore will give a much better measure of generalizability. The full procedure is outlined in the pseudo-code on the next page.

```
LEAVE-ONE-GROUP-OUT-CROSS-VALIDATION(PrimerDataset, Groups)
 1: Let hp denote GPT-2's hyperparameters
 2: for group in Groups do
 3:     QueryDataset = PrimerDataset rows containing the field "group"
 4:     PrimerDataset = PrimerDataset rows not containing the field "group"
 5:     for each Query in QueryDataset do
 6:         for each PRIMER-SELECTION-METHOD do
 7:             PrimerSet = PRIMER-SELECTION-METHOD(Query, PrimerDataset)
 8:             for p = 1 to number of permutations do
 9:                 permute PrimerSet
10:                 GPT2_input = PrimerSet || Query
11:                 reflection = GPT2(GPT2_input, hp)
12:                 label = RQC(Query, reflection)
13:                 append reflection and label to the relevant data-structures
14:             end for
15:         end for
16:     end for
17: end for
```

4.5 Primer Dataset Expansion

Consider how one would implement a primer-selection method in practice. They would obtain a primer dataset and then deploy the model for their application. After the model is used many times, the developer may collect good examples of prompt/response/reflection triples, and might want to update their primer dataset to include these examples. The question becomes, which primer-selection method will preform the best as one adds more primers to their primer dataset?

The method I propose for determining this is called *primer dataset expansion*. Suppose we have an external dataset of queries (prompt/response pairs) that is different than primer dataset. The idea is to start with an empty primer dataset, and iteratively add primers to this primer dataset. In each iteration, we generate a reflection and corresponding label for each query in the query dataset. The process repeats until we have added every primer to the primer dataset. We can then analyze how the "Good Reflection Average" changes as the number of primers in the primer dataset increases.

This procedure could be done by adding one primer at a time; however, this would take far too long. Instead, groups of primers will be added according to the Type and Topic groupings from Section 4.1. This procedure is outline in the pseudo-code on the next page.

```
PRIMER-DATASET-EXPANSION(PrimerDataset, QueryDataset, Groups)
Let hp denote GPT-2's hyperparameters
initialize ExpandingPrimerDataset as empty data-structure
for group in Groups do
    append PrimerDataset rows containing field group to ExpandingPrimerDataset
    for each Query in QueryDataset do
        for each PRIMER-SELECTION-METHOD do
            PrimerSet = PRIMER-SELECTION-METHOD(Query, ExpandingPrimerDataset)
            for p = 1 to number of permutations do
                permute PrimerSet
                GPT2_input = PrimerSet || Query
                reflection = GPT2(GPT2_input, hp)
                label = RQC(Query, reflection)
                append reflection and label to the relevant data-structures
            end for
        end for
    end for
end for
```

One issue is that we do not have an external query dataset due to the limited amount of data. Recall from Section 4.1 that there are actually two primer datasets, one containing simple reflections and the other containing complex reflections. In order to approximate an external query dataset, when testing the simple-reflection primer dataset the query dataset will be the complex-reflection primer dataset, and vice versa. This isn't ideal, since many of the prompt/response pairs are very similar between the primer datasets. However, the purpose of this experiment is to measure the effect of increasing the primer dataset and not the generalizability. Therefore, when analyzing these results we must keep in mind that it is the overall trend that we are interested in, and not the "Good Reflection Average" value at any particular iteration.

5 Results and Discussion

5.1 Hyperparameter Grid Search

Tables 3 and 4 contain the results of the hyperparameter grid search procedure from Section 4.3 for both the simple-reflection and complex-reflection primer dataset.

The column "Good Reflection Average" represents the average number of reflections labeled as good by the RQC. Recall from Section 4.3.2 that multiple reflections were generated for each query by permuting the chosen primers. The column "Average Within Example Standard Deviation" represents the average of the standard deviation of the RQC scores of the reflections generated with the same prompt/response pair. Therefore, this column measures the variance caused by permuting the chosen primers. The column "Average Between Example Standard Deviation" represents the standard deviation of the average of the RQC scores reflections generated with the same prompt/response pair. Therefore, this column measures the variance caused by the different prompt/response pairs. In other words, the difference between the last two columns is the order in which the standard deviation and the average of the RQC scores are taken.

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Greedy	0.847	0.138	0.286
Sampling - Low Variance	0.880	0.110	0.255
Sampling - Med Variance	0.867	0.165	0.235
Sampling - High Variance	0.493	0.401	0.296

Table 3: Hyperparameter Results for the Simple-Reflection Primer Dataset

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Greedy	0.350	0.158	0.411
Sampling - Low Variance	0.923	0.105	0.169
Sampling - Med Variance	0.849	0.203	0.217
Sampling - High Variance	0.549	0.419	0.280

Table 4: Hyperparameter Results for the Complex-Reflection Primer Dataset

It is clear from the data that the "Sampling - Low Variance" row produces the best results, as it has the highest "Good Reflection Average" and the lowest "Average Within Example Standard Deviation" and "Average Between Example Standard Deviation". Therefore, the hyperparameters used in GPT-2 for the following results are a temperature of 0.1 and top k of 10.

5.2 Leave-One-Group-Out Cross Validation

The following are the results of the "Leave-One-Group-Out Cross Validation" procedure described in Section 4.4 for each primer selection method (random, similar, and different), for each group (Topic and Type), and for both the simple and complex primer datasets.

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Random	0.778	0.247	0.242
Similar	0.768	0.310	0.235
Different	0.522	0.210	0.465

Table 5: Leave-One-Topic-Out Cross Validation for the Simple-Reflection Primer Dataset

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Random	0.579	0.264	0.397
Similar	0.659	0.227	0.395
Different	0.356	0.179	0.468

Table 6: Leave-One-Topic-Out Cross Validation for the Complex-Reflection Primer Dataset

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Random	0.804	0.234	0.261
Similar	0.796	0.266	0.240
Different	0.519	0.305	0.221

Table 7: Leave-One-Type-Out Cross Validation for the Simple-Reflection Primer Dataset

	Good Reflection Average	**Average Within-Example Standard Deviation**	**Average Between-Example Standard Deviation**
Random	0.564	0.291	0.399
Similar	0.665	0.305	0.315
Different	0.254	0.250	0.315

Table 8: Leave-One-Type-Out Cross Validation for the Complex-Reflection Primer Dataset

From these results, it is clear that the "different" primer-selection method is worse than "random" and "similar" as it generally has the lowest "Good Reflection Average" by a large margin.

Furthermore, there does not seem to be a clear difference in generalizability between the "random" and "similar" methods based on these results as all column values are within a reasonable margin. The next experiment will highlight the difference between these methods.

5.3 Primer Dataset Expansion

Figures 10 and 11 on the next page show the plots of the primer dataset expansion procedure as described in Section 4.5. As the plot moves to the right, groups are being added to the primer dataset. Note that this procedure was only preformed on the Type grouping and not the Topic grouping due to the significant amount of computation required to run this analysis.

The important thing to keep in mind while looking at these plots is that the overall trend of the methods are more important than the precise value of the "Good Reflection Average" at any particular iteration. As discussed in Section 4.5, the query dataset and primer dataset are very similar. Therefore, the "Good Reflection Average" cannot be interpreted as a measure of generalizability as it is in the leave-one-group-out cross validation evaluation method. Instead, we are interesting in whether the primer-selection methods tend to increase, decrease, fluctuate, or plateau as the number of groups in the primer dataset increases.

Figures 10 and 11 show that the random primer-selection method tends to fluctuate as the number of groups in the primer dataset increases. Random reaches both the highest and lowest "Good Reflection Average" value in both plots both a large margin. By contrast, similar and different have very few fluctuations. The similar primer-selection method stayed mostly the same in Figure 10, and in Figure 11 increases until a certain point before plateauing. The different primer-selection method had similar behavior to the similar primer-selection method in Figure 10, but had the opposite behavior in Figure 11 where it decreases until a certain point before plateauing.

Figures 10 and 11 reinforces the results from the leave-one-group-out cross validation in Section 5.2, showing that the different primer-selection method is worse than the similar primer-selection method. Furthermore, these results show that the similar primer-selection method is a more reliable method than random primer-selection due to much smaller fluctuations in performance, and its tendency to either increase or plateau as the number of primers increases.

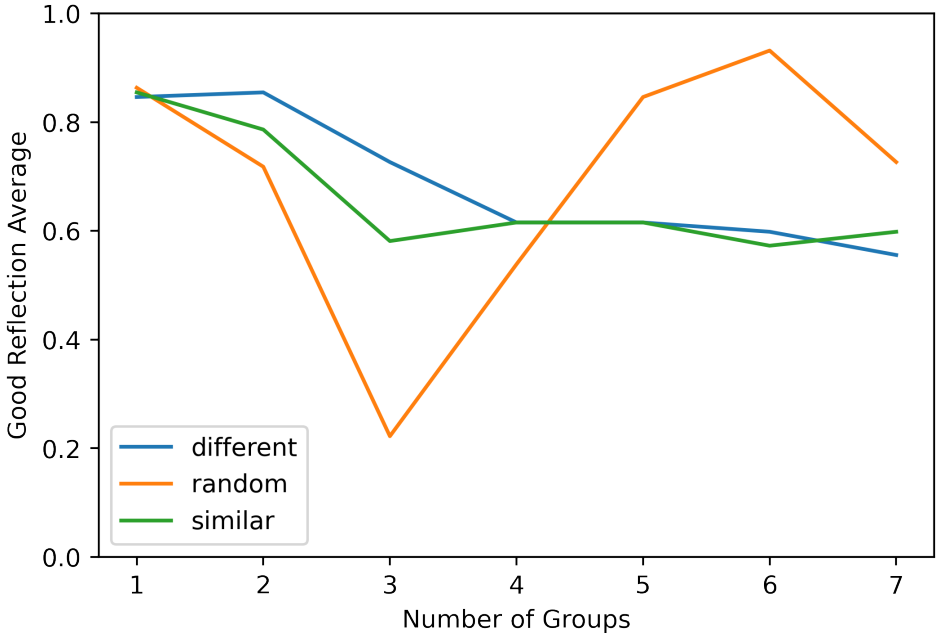

Figure 10: Primer Set Expansion using the Simple-Reflections Primer Dataset

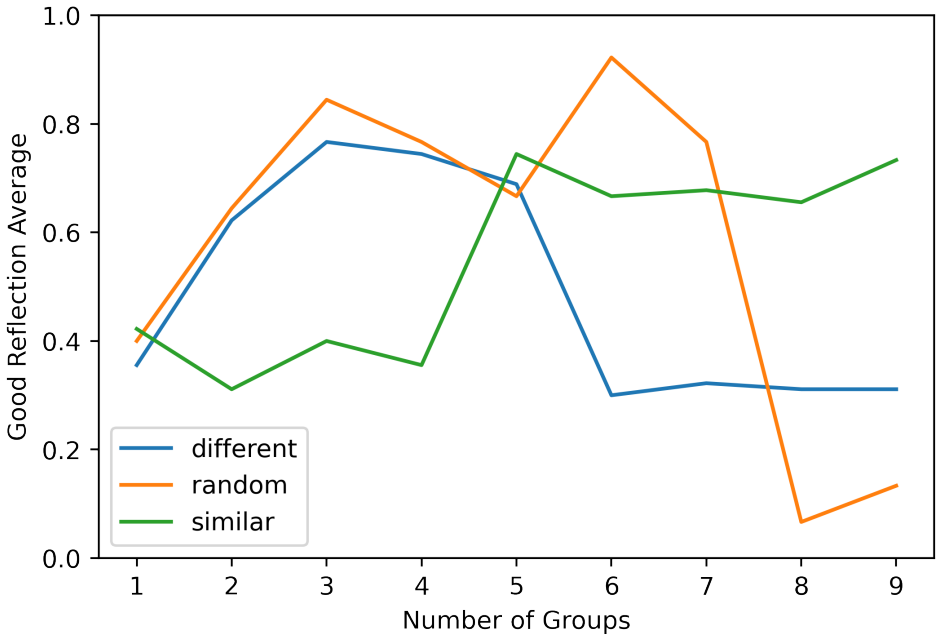

Figure 11: Primer Set Expansion using the Complex-Reflections Primer Dataset

6 Conclusion

Few-shot learning is a very important and powerful tool which allows GPT-2 the ability to learn tasks with very few examples. The method involves pre-pending examples, called primers, of the task correctly done to the beginning of the input. However, prior to this thesis, there was very little research as to how one should select these primers. The contribution of this thesis is to propose and evaluate various primer-selection methods.

The results of this research suggests that choosing semantically similar primers to the query tends to produce more reliable and robust reflections compared to the other proposed methods. This shows that in the context of generating reflections a primer-selection method exists that preforms better than selecting primers at random. This provides a stepping-stone for future research in creating more sophisticated primer-selection methods for improving the quality of any text-generation task.

6.1 Future Work

The first avenue one should take to move this research forward is to obtain a higher quantity of data and a more diverse dataset. Fahad's dataset is very self-similar, which is why the leave-one-group-out cross validation technique was required. In order to claim definitively that one primer-selection method preforms better than another, more and better data is necessary.

Another avenue one may take is to fine-tune or re-train GPT-2 and the universal sentence encoder on Motivational Interviewing text. Currently, these models are pre-trained on general text scraped indiscriminately from the World Wide Web. It is very unlikely that Motivational Interviewing makes up a significant portion of this training corpus. Therefore, fine-tuning or re-training these models on text more relevant to the generation task should significantly improve their performance.

Finally, the last avenue one might take is to improve on the proposed primer-selection methods. This research does not claim that the similar primer-selection method is the best possible primer-selection method, rather that it preforms better than the other proposed methods. Looking at Figure 10 and 11, at some points the random primer-selection method performs better than the similar primer-selection method. This suggests better primers could have been chosen compared to those chosen by the similar primer-selection method. One possibility is to quantify the concept of "diversity" and select the primers that are both similar to the query, but also diverse with respect to each other. It is well known that diversity in training data leads to better generalizability, and that may also be the case with few-shot learning.

References

[1] S. Rollnick and W. R. Miller, *Motivational interviewing*. The Guilford Press, 2013.

[2] D. T. Lai, K. Cahill, Y. Qin, and J.-L. Tang, "Motivational interviewing for smoking cessation," *Cochrane Database of Systematic Reviews*, 2010.

[3] F. Almusharraf, "Motivating smokers to quit through a computer-based conversational system," Master's thesis, University of Toronto, 2018. unpublished thesis.

[4] Center for Disease Control and Prevention, "Smoking cessation: Fast facts," 2020.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[7] A. Maseeh and G. Kwatra, "A review of smoking cessation interventions," *Medscape General Medicine*, vol. 7, p. 24, June 2005. PMID: 16369403.

[8] U.S. Department of Health and Human Services, "The health benefits of smoking cessation," U.S. Department of Health and Human Services, Public Health Service, Centers for Disease Control, Center for Chronic Disease Prevention and Health Promotion, Office on Smoking and Health. DHHS Publication No. (CDC) 90-8416, 1990.

[9] M. E. Wewers, F. A. Stillman, A. M. Hartman, and D. R. Shopland, "Distribution of daily smokers by stage of change: Current population survey results," vol. 36, pp. 710–20, June 2003. PMID: 12744915.

[10] Centers for Disease Control and Prevention (CDC), "Cigarette smoking among adults–united states, 2006," vol. 56, pp. 1157–61, November 2007. PMID: 17989644.

[11] R. Rosenfeld, "Two decades of statistical language modeling: where do we go from here?," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[12] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*,

(Prague, Czech Republic), pp. 858–867, Association for Computational Linguistics, June 2007.

[13] J. Ba, "CSC421/2516 Lecture 3: Automatic Differentiation & Distributed Representations." unpublished lecture slides, 2020.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2018.

[16] J. Vuckovic, A. Baratin, and R. T. des Combes, "A mathematical theory of attention," 2020.

[17] J. Alammar, "The Illustrated GPT-2 (Visualizing Transformer Language Models)," 2019.

[18] Huggingface, "How to generate text: using different decoding methods for language generation with Transformers." unpublished google colab notebook, 2020.

[19] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," 2018.

[20] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2020.

[21] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," 2021.

Appendices

A Simple and Complex Reflection Comparison

A *reflection* is a statement intended to mirror what has been previously said. A *simple reflection* contains little or no additional content, and is similar to a restatement. A *complex reflection* infers additional meaning beyond what the client has said.

The following is a small example to illustrate the difference between a simple and complex reflection.

A.1 Simple Reflection Example

> `Client`: I think I'm probably being too careful. My last test results were good. It just scares me when I feel pain like that.
>
> `Interviewer`: You are feeling scared.

A.2 Complex Reflection Example

> `Client`: I think I'm probably being too careful. My last test results were good. It just scares me when I feel pain like that.
>
> `Interviewer`: It reminds you of your heart attack.

A.3 Comparison

The simple reflection is very surface level. The interviewer has essentially repeated back what the client has said. While these types of reflections can have value in the right context, if employed too often, clients will start to get aggravated.

In the complex reflection example, the interviewer does two things differently. First, the interviewer is making a connection from earlier in the conversation or even from a past engagement. Second, the interviewer is making an inference or an educated guess as to how the client is feeling. These reflections, when done correctly, can be very powerful.

B Word2Vec Architectures

The following are more in-depth diagrams of what was proposed in the original Word2Vec paper. Note that the gray ellipsoid shapes represent a set of nodes [14].

B.1 Continuous Bag of Words

The input nodes (Token $t-1$, Token $t-2$, etc) are a series of one-hot encoded nodes. The word vectors are contained in the weight matrix W^T, as the one-hot encoded vector will simply pick out a row of matrix W.

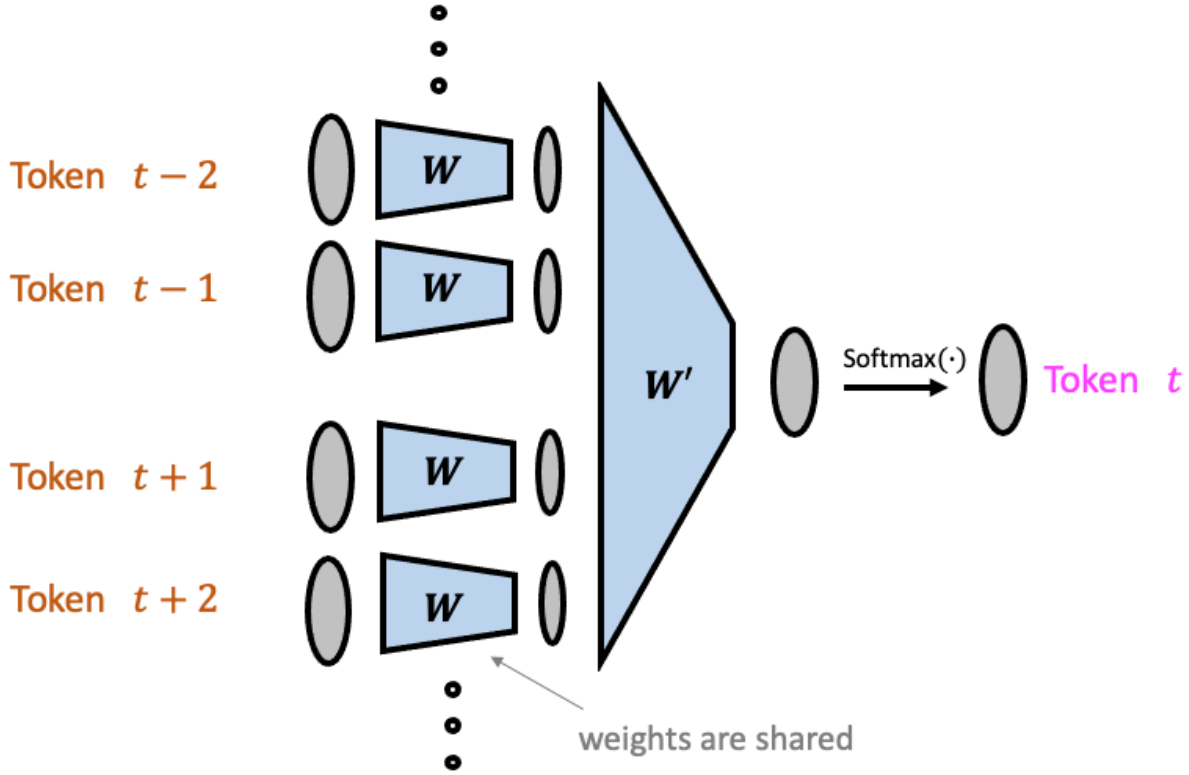

Figure 12: Continuous Bag of Words Architecture

B.2 Skip Gram

The input nodes (Token $t-1$, Token $t-2$, etc) are a series of one-hot encoded nodes. The word vectors are contained in the weight matrix W^T.

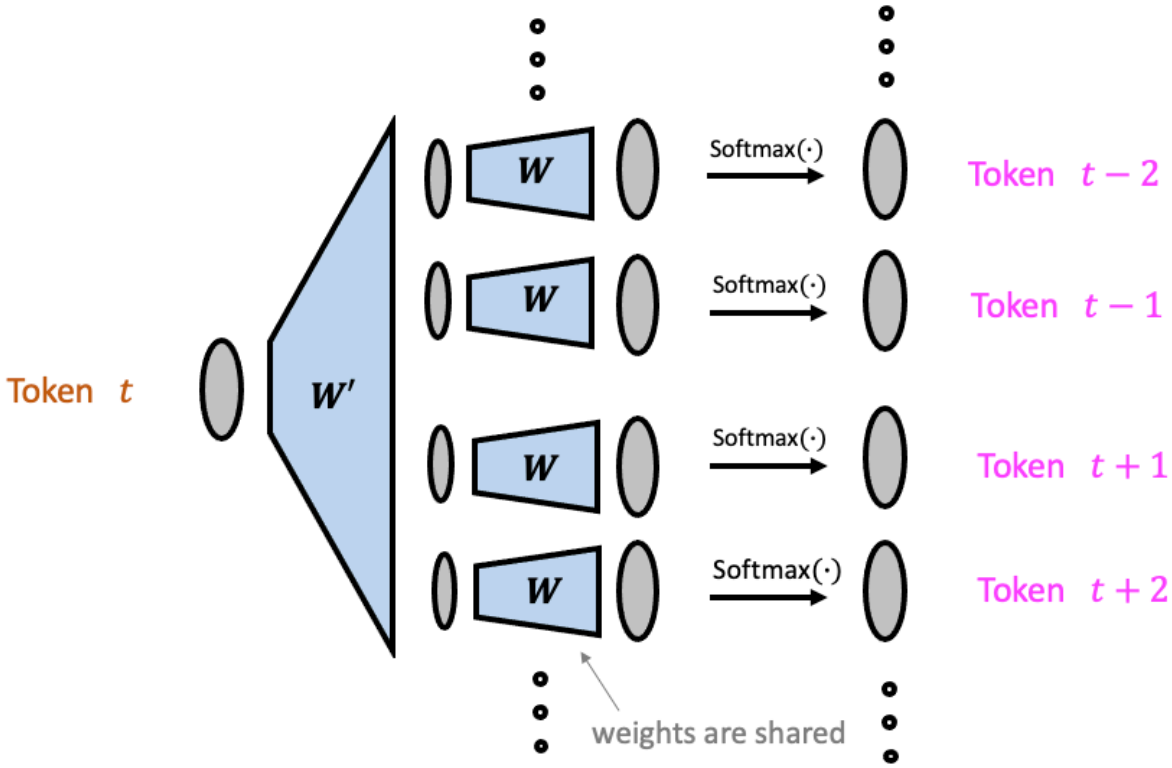

Figure 13: Skip Gram Architecture

In practice, the skip-gram model tends preform better than continuous bag-of-words. However, it requires more data in order to achieve better performance.

C Attention Networks

These are the attention mechanism originally proposed by the seminal paper "Attention is All You Need". Q, K, and V stand for query, key, and value respectively. The query is the word you are analyzing, and the key and the value are the sequence you are interested in [5]. Note, query in this context is referring to something completely different than the query of GPT-2's input.

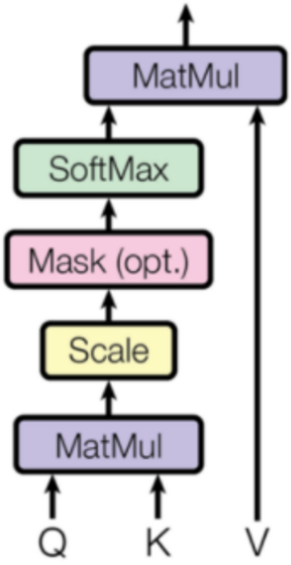

Figure 14: Scaled Dot Attention

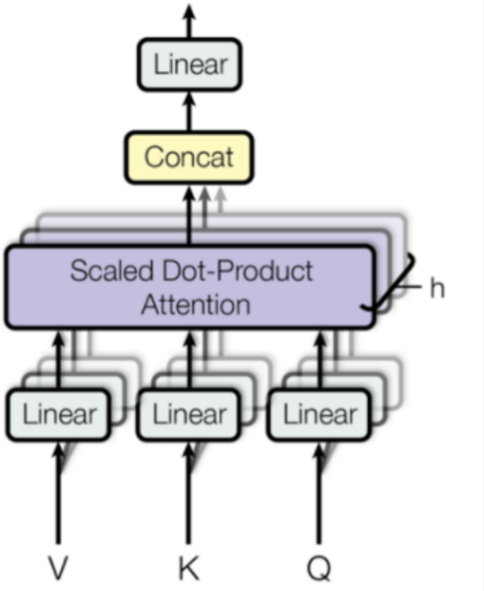

Figure 15: Multi-Head Attention

D Cosine Similarity and Euclidean Similarity

When you embed text into a latent vector space, it is generally accepted that cosine similarity gives you a better measure of semantic similarity compared to Euclidean similarity. Let's see why.

D.1 Equations

Intuitively, a similarity metric is like an inverse of a distance metric. In other words, as distance approaches 0, similarity approaches 1 and as distance approaches it's maximum value, similarity approaches 0. We can write the Euclidean similarity and cosine similarity as the following.

$$\text{euclidean\_similarity}(\boldsymbol{v}, \boldsymbol{u}) = \frac{1}{1 + d(\boldsymbol{v}, \boldsymbol{u})} = \frac{1}{1 + \sqrt{\boldsymbol{v}^T \boldsymbol{u}}} \tag{9}$$

$$\text{cosine\_similarity}(\boldsymbol{v}, \boldsymbol{u}) = 1 - \frac{\boldsymbol{v}^T \boldsymbol{u}}{|\boldsymbol{v}||\boldsymbol{u}|} \tag{10}$$

D.2 An Example

Consider the following example shown in Figure 16. Here, we would like to determine which of the red vectors is more similar to the blue vector. The Euclidean similarity would say the bottom-most vector is more similar, while cosine similarity would say the longest vector is the most similar.

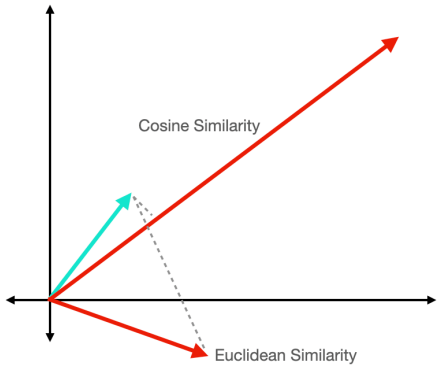

Figure 16: Euclidean Similarity vs Cosine Similarity Example

Empirically, it turns out that Cosine Similarity is correct in this example. Often a vector being longer in one dimension means it contains more of the semantic meaning it is encoding. Whereas the vector chosen by the Euclidean Similarity will contain semantic meaning that is completely different. Thus, intuitively, this is why we prefer the cosine similarity measure compared to the Euclidean similarity measure.