UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS num. 2996

Personality prediction from text using prompt-based learning

Rino Čala

Zagreb, June 2022.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 11 March 2022

MASTER THESIS ASSIGNMENT No. 2996

Student:	**Rino Čala (0036505532)**
Study:	Computing
Profile:	Computer Science
Mentor:	prof. Jan Šnajder

Title: **Personality prediction from text using prompt-based learning**

Description:

Personality is a characteristic feature that makes an individual unique in a population and that manifests as a set of opposing personality traits. Because an individual's personality is one of the fundamental characteristics of humanity and is essential to recognizing and predicting human behavior, machine learning-based approaches to personality prediction are of theoretical and practical importance. In the field of natural language processing, a number of models for predicting an author's personality from text have been proposed and obtained promising results. The topic of this thesis is the prediction of personality traits based on a novel deep learning approach for natural language processing that uses prompt-based learning. Study the existing models for deep-learning-based personality prediction from text and prompt-based learning approaches. Select a number of such models, implement them, and apply them to the task of personality prediction on publicly available data. Perform hyperparameter optimization and a thorough analysis of the effectiveness of different templates (both manually created and learned from data), training duration, and prediction generation modes. Also, analyze how the performance of the model depends on the amount of data available. Perform a statistical analysis of the results obtained. All references must be cited, and all source code, documentation, executables, and datasets must be included with the thesis.

Submission date: 27 June 2022

Hvala obitelji i bližima što su mi bili podrška na ovom putu.
Hvala mentoru Janu Šnajderu i Josipu Jukiću na prijedlozima i pomoći tijekom izrade rada

CONTENTS

1. Introduction

Artificial intelligence is one of the most rising research fields today dealing with developing and understanding machine intelligence. The area is widespread and considers numerous applications that are well established in society today, which few of them are advanced web search engines, human speech understanding, self-driving vehicles, recommendation systems, chatbots, and others. Considering the size of the field, numerous subfields have been established under which experts focused their efforts on developing architectures for specific applications.

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, mainly how to program computers to process and analyze large amounts of natural language data. When building intelligent machine learning systems that could understand natural language, standard practice was to create hand-made rules or features on large quantities of annotated and unannotated texts. After creating useful features from texts, traditional machine learning models were used to develop a useful model which could solve a nontrivial task. As neural networks have emerged as a method that successfully solved numerous tasks with state-of-the-art results, natural language processing has begun to lean toward neural approaches.

Recurrent neural network (RNN) (Rumelhart and McClelland, 1987) was the first neural network architecture that could capture the extended context of sentences. As the RNN model struggled with vanishing gradients and loss of context over long sentences, there was a need for a better recurrent architecture. Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) suffered much less from vanishing gradients and it behaved much better over long sentences with the novel introduction of input, output, and forget gate for better handling of knowledge embedded into the hidden state of the model. Gated recurrent unit (GRU) (Chung et al., 2014) was built upon the ideas of the LSTM model, introducing update and reset gates and the overall performance of the model was similar to the LSTM model. When feeding words of sentences to the machine learning models, they should be represented with vectors that capture their ingrained meaning. Continuous Bag of Words (CBOW) model (Mikolov et al., 2013a) and Skip-gram (Mikolov et al., 2013b) model were

two-layer neural network models which developed word representations during training with different objectives. CBOW model maximized the probability of center word from surrounding context words. On the other hand, the Skip-gram model maximized surrounding context words while having the center word. Vectors produced during the training of the models have been shown to capture the word's context accurately, and related words appear closer to each other in d-dimensional space.

Introduced models captured the context of the sentence, but dependencies and correlation between words in the sentence still could not be captured fully. The latest Transformer architecture (Vaswani et al., 2017) could capture dependencies of the words while preserving context over long sentences. Transformer architectures have obtained state-of-the-art results on almost all NLP tasks and still are a dominant architecture used in developing solutions for natural language tasks. To provide better initialization of the model, several pretraining procedures were carried out over large corpora of text, which resulted in high performing pretrained language models, like BERT (Devlin et al., 2018), GPT2 (Radford et al., 2019), ALBERT (Lan et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), T5 (Raffel et al., 2019) and other. These pretrained models were additionally modified and finetuned on downstream tasks obtaining state-of-the-art results.

Larger and deeper models have been shown to obtain better results than the more shallow ones, so the latest research concentrated on developing models that have moved from millions to billions of parameters. One of the first deeper models was a GPT3 model (Brown et al., 2020) with 175 billion parameters. Large natural language models are hard to fine-tune on downstream tasks. Still, because of the more extensive ingrained knowledge, models have excellent performance in few-shot or zero-shot setting and in natural language generation and conversation tasks. Training the models is expensive, and lack of annotated data is common, so large architectures like these could help avoid these problems in practice.

Task performance of large language models heavily depends on how the input is fed to the model. To provide better guidance to the model, prompt-based learning has emerged, dealing with the creation of textual or continuous prompts that the model is tasked to fill with the downstream task's target text. Textual prompts are usually designed by hand and are tested by measuring the model's performance on a downstream task. Creating well-performing prompts is challenging, but several hand-made prompts were created that perform very well. To avoid the difficulty of manually creating prompts, continuous prompts were introduced. Continuous prompts rely on newly added tokens appended to downstream tasks and are usually fine-tuned solely without fine-tuning the whole model. This approach showed great results while eliminating the process of manual prompt creation and training of large language models.

Prompts with pretrained language models have been used on most common natural language tasks, but personality prediction is a task that they have not been used on. Personality prediction predicts a person's personality, a distinct set of behaviors, cognitions, and emotional patterns that evolve from biological and environmental factors. There are several theories on dimensions that a person's personality is built upon. Relying on different theories, several personality tests were introduced. Myers-Briggs Type Indicator (MBTI) is a standardized test of a person's personality which states that a personality is made of four opposite personality trait pairs: introverted/extraverted, sensing/intuitive, thinking/feeling, judging/perceiving. A person's personality is a crucial human dimension that can be used in several applications, like targeted marketing and increasing user retention. A personality dataset was built called PANDORA (Gjurković et al., 2020) to enable the building of machine learning models that could predict personality. PANDORA dataset consists of comments and personality traits of Reddit users with other characteristic human dimensions, like gender, age, and others. With the release of the dataset, it became possible to build machine learning models that will have high performance in predicting human personality dimensions.

I used the PANDORA dataset to build language models that will predict human personality dimensions. After filtering the dataset with the TinyBERT model, only comments that contained useful personality indicators were left. Firstly, I trained BERT and GPT2 models to predict a person's opposite personality traits from comments in a classic few-shot and full data binary label setting. After that, I used the same models with five hand-made prompts and continuous prompts in a zero, few, and full data setting, tasking the model to fill the prompts with person's opposite personality traits. I managed to produce high-performing models using prompts that proved to provide similar performance or outperform models trained in classic data setting. By tuning only continuous prompts and keeping the model's parameters frozen, with just a fraction of tuned parameters, I managed to obtain high performance of the models. Hand-made prompts used in this work could be used when training personalized personality prediction models or as a good start for manual or automatic prompt search.

2. Models

2.1. Transformer

Transformer (Vaswani et al., 2017) is a novel architecture displayed in Figure 2.1, which relies on attention mechanism. It consists of an encoder that encodes the input sentence and a decoder that takes in the outputs of previous transformer steps. When feeding the input words to the model, they are first embedded into an embedding vector of size $d_{model} = 512$. To consider the positions of words in the sentence, the authors used positional embeddings, which are calculated according to expressions (2.1) and added to the word embeddings. For words that appear at index pos in the sentence, for even places $2i$ in the positional embedding, we use the formula with the sine, and for the odd places $2i + 1$ in the positional embedding, we use the formula with the cosine.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \end{aligned} \tag{2.1}$$

Encoder part of the model consists of stacked identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network. A residual connection is applied around each of the sub-layers with layer normalization.

Decoder is also composed of stacked identical layers like the encoder part. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, residual connections are employed around each sub-layer, followed by layer normalization. To prevent the position i from attending to subsequent positions of i, the self-attention sub-layer is modified to attend only to previous words, and output embeddings are offset by one position.

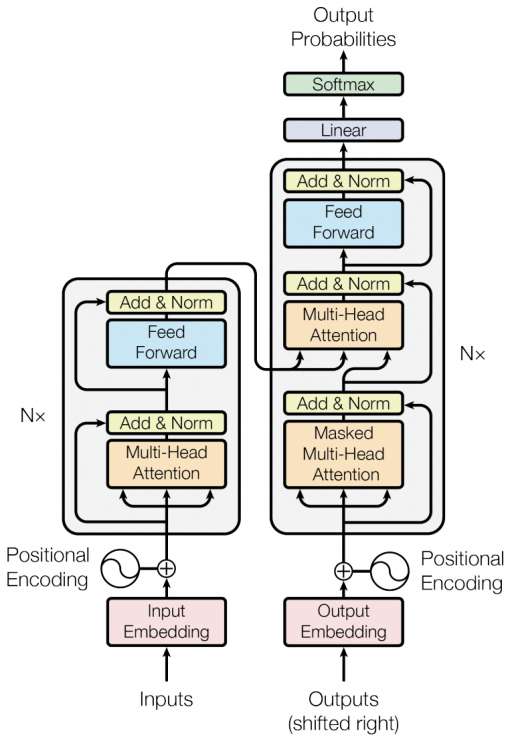

Figure 2.1: Transformer architecture (Vaswani et al., 2017)

In the multi-head attention layers of the model, the authors used the attention mechanism according to the expression (2.2). In the self-attention sub-layers of the model, queries, keys, and values correspond to the transformed inputs to the sub-layer, while in the cross-attention sub-layers, queries correspond to the transformed decoder inputs and keys and values to the transformed outputs of the encoder stack. The transformation before the attention mechanism is well depicted in Figure 2.2. After the attention mechanism is performed for every head of the multi-head attention, outputs are concatenated into a unique vector of size d_{model}.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

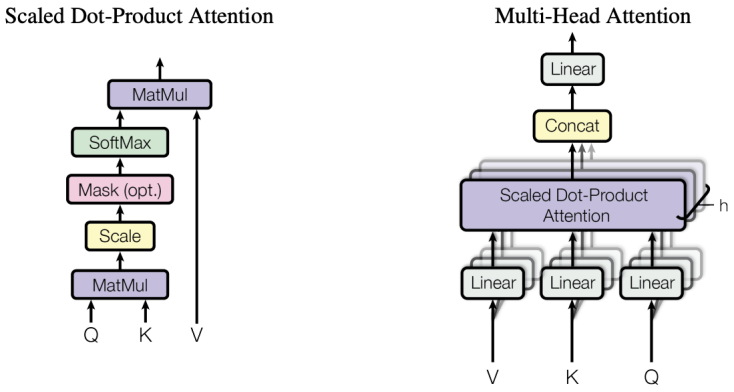

Figure 2.2: Multi-head attention (Vaswani et al., 2017)

2.2. BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) is a bidirectional multi-layer transformer encoder model which builds upon the ideas of Vaswani et al. (2017). For better model parameter initialization, the authors introduced a novel pre-training objective called masked language modeling (MLM). They fed the sentences of the pretraining corpus to the model but masked a proportion of the words and tasked the model to predict them at the output. They chose 15% of words at random and replaced them with a [MASK] token 80% of the time, random word 10% of the time, and did not change the word 10% of the time. They also introduced a next sentence prediction task (NSP). The model is fed two subsequent sentences or non-subsequent sentences in the pretrain corpus and is tasked to predict if they naturally come one after the other. For the pretraining corpus, the authors used Wikipedia (2500M words) and BooksCorpus (800M words).

When feeding the sentences to the model, three types of embeddings were used, which are simply added into a unique embedding. Words are represented with the Wordpiece embedding vocabulary with embeddings for 30000 english tokens. To take into account the word's position in the sentence, positional embeddings were used. A single word belongs to the first or second sentence concatenated by the [SEP] token in the NSP task, so the authors also used segment embeddings. The creation of input representation is shown in Figure 2.3. Before feeding the input to the model, [CLS] token is added at the beginning for the NSP task, and later, after finetuning, for the downstream classification and similar tasks.

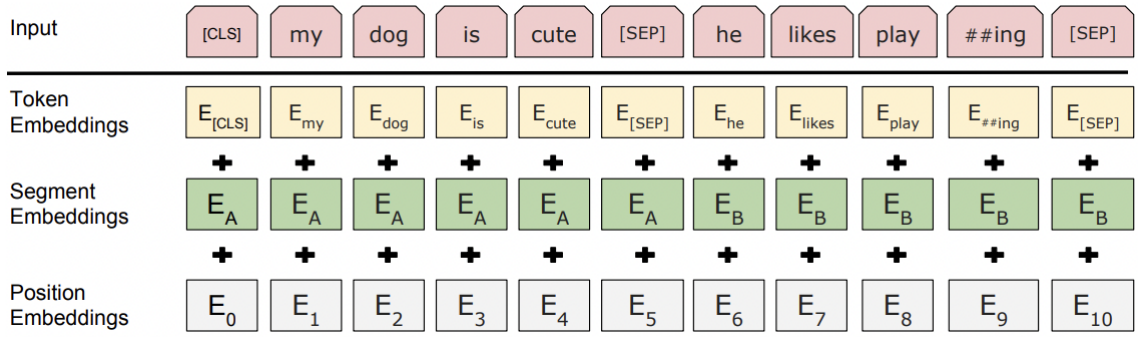

Figure 2.3: BERT input representation (Devlin et al., 2018)

Input goes through L layers of the transformer model and is represented at the output as transformed hidden states. Hidden states of masked words and [CLS] token are used in the masked language modeling and NSP tasks. The authors have used two model sizes. The smaller model is BERT$_{BASE}$ with L=12 layers, A=12 self-attention heads, and the size of hidden states is H=768 with a total of 110M parameters. The larger model is BERT$_{LARGE}$ with L=24, H=1024, and A=16 with a total of 340M parameters. After pretraining the model

on the pretrain corpus, the model is used on several downstream tasks acquiring state-of-the-art results. The whole procedure of pretraining and finetuning of the model is shown in Figure 2.4

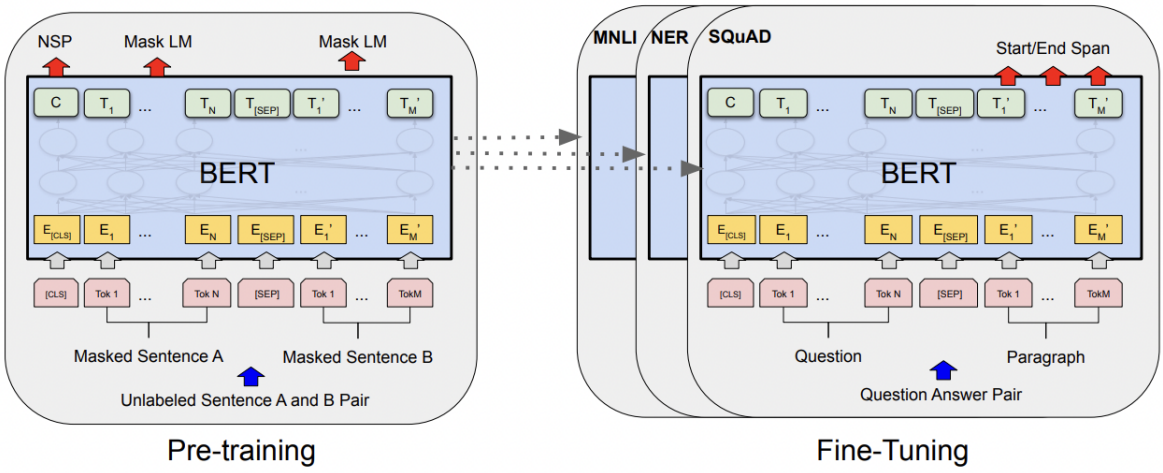

Figure 2.4: BERT pretraining and finetuning (Devlin et al., 2018)

2.3. GPT2

GPT2 model (Radford et al., 2019) is a unidirectional multi-layer transformer decoder model which uses a language modeling task to pretrain the parameters of the transformer architecture. During the model's training, the probability of the next word in the sentence is maximized by inputting the previous word and conditioning on the words that appear earlier in the sentence. The probability that is maximized is shown in (2.3). Because of its unidirectional approach, words can only attend to words that come before them in the sentence.

$$p(x) = \prod_{i=1}^{n} p\left(s_n \mid s_1, \ldots, s_{n-1}\right) \tag{2.3}$$

To pretrain the transformer architecture, the authors created a new web scrape that emphasizes document quality. They sampled web pages from links mentioned in a post with at least three karma on Reddit followed by human curating. The resulting dataset WebText contains the text subset of these 45 million links with approximately 8 million documents and 40 GB of text.

The authors wanted to use a practical middle ground between character and word-level language modeling, settling on using byte pair encoding (BPE). Transformer model was incorporated with just a few changes. Layer normalization was moved to the input of each sub-block, similar to a pre-activation residual network, and an additional layer normalization was added after the final self-attention block. The vocabulary was expanded to 50257 tokens, and the max input size was increased from 512 to 1024 tokens. They used a transformer architecture with L=12 layers, a hidden state size of H=768, and A=12 attention heads.

3. Evaluation metric

Evaluation metrics are used to examine the performance of created models on a specific task and choose the best performing model. This work will use the F1 metric to evaluate the models and approaches. F1 is calculated as a harmonic mean between precision and recall according to (3.1). The value ranges from 0 to 1, where a higher value means a better performing model.

$$F1 = \frac{2 \times \text{ precision } \times \text{ recall}}{\text{precision } + \text{ recall}} \tag{3.1}$$

Precision is calculated according to (3.2). True positives (TP) are examples that were classified as positive, and their true class label is also positive. False positives (FP) are examples classified as positive, but their true class label is negative. The metric calculates the percentage of positively predicted examples that are positive.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

Recall is calculated according to (3.3). False negatives (FN) are examples classified as negative but actually of the positive class. The metric calculates the percentage of true positive examples that were retrieved as positive.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

In this work, a macro F1 value is used, calculated by averaging the F1 value for each class considered as a positive class and the rest of the classes as negative.

4. Prompt-based learning

In traditional supervised learning for NLP, an input text $\mathbf{x}$ is taken and an output $\mathbf{y}$ is predicted according to the conditional probability $P(\mathbf{y}|\mathbf{x};\theta)$, where θ corresponds to model parameters. In order to learn the parameters θ, a dataset containing pairs of inputs and outputs is used during the training procedure. The goal of the training procedure is to maximize the probability of true outputs having fed the corresponding input to the model. The main issue with supervised learning is that in order to train a model $P(\mathbf{y}|\mathbf{x};\theta)$, it is necessary to have supervised data for the task, which cannot be found in large amounts. Prompt-based learning (Liu et al., 2021) for NLP attempts to circumvent this issue by modeling the probability $P(\mathbf{x};\theta)$ of text $\mathbf{x}$ itself, avoiding the need for large supervised datasets.

When forming a prompt around the input text, a prompting function f_{prompt} is used to transform the text into a prompt $\mathbf{x'} = f_{prompt}(\mathbf{x})$. The creation of the prompt is consisted of two steps:

1. A textual template is used with an input slot $[X]$ and an answer slot $[Z]$. Answer slot is used for generated prompt filling $\mathbf{z}*$ that is mapped to a true class $\mathbf{y}*$.

2. After the template is constructed, the $[X]$ slot is filled with the input text $\mathbf{x}$.

Depending on the position of the answer slot $[Z]$, there are two types of prompts. Prefix prompts have an answer slot at the very end of the prompt, while cloze prompts have an answer slot positioned somewhere in the middle. During the training of the model, we define $\mathcal{Z}$ as a set of permissible values for $\mathbf{z}$ from which the answer slot $[Z]$ will be filled. For example, a classification task could have $\mathcal{Z}$ defined as $\mathcal{Z}$ = {"excellent", "good", "OK", "bad", "horrible"} to represent each of the true classes in $\mathcal{Y}$ = {++, +, , -, −}. The trained model is used for inference by filling the prompt with a generated answer $\mathbf{z}^*$ which is then mapped to the predicted label $\mathbf{y}^*$. The full terminology and an example of prompt creation are shown in Table 4.1

Table 4.1: Prompt creation

Name	Notation	Example	Description
Input	**x**	I love this movie.	One or multiple texts
Output	**y**	++ (very positive)	Output label or text
Prompting Function	$f_{prompt}(\mathbf{x})$	$[X]$ Overall, it was a $[Z]$ movie.	A function that converts the input into a specific form by inserting the input **x** and adding a slot $[Z]$ where answer **z** may be filled later.
Prompt	**x'**	I love this movie. Overall, it was a $[Z]$ movie.	A text where $[X]$ is instantiated by input **x** but answer slot $[Z]$ is not.
Filled Prompt	$f_{fill}(\mathbf{x'}, \mathbf{z}^*)$	I love this movie. Overall, it was a bad movie.	A prompt where slot $[Z]$ is filled with any answer.
Answered Prompt	$f_{fill}(\mathbf{x'}, \mathbf{z})$	I love this movie. Overall, it was a good movie.	A prompt where slot $[Z]$ is filled with a true answer.
Answer	**z**	"good", "fantastic", "boring"	A token, phrase, or sentence that fills $[Z]$

During prompt engineering, there are different design considerations that go into a prompting method:

- Pre-trained model choice
- Prompt Engineering
- Answer Engineering

Prompt engineering is the process of creating a prompting function $f_{prompt}(\mathbf{x})$ that will result in high performance on a downstream task. There are two varieties of prompts previously mentioned, cloze prompts and prefix prompts. Which one will be used depends on the task and language model being used. In general, cloze prompts will be used for tasks that are solved using masked language modeling. When generation models are used, a prefix prompt is a good choice. Finding the best prompt for a task is a hard problem requiring ex-

tensive search and testing of model performance. The most natural way to create prompts is to manually develop intuitive templates based on human introspection. This search method is a sort of art that takes time and experience where even experienced prompt designers fail to discover optimal prompts manually. To avoid the problems of manual search, an automated template search is incorporated. Automatic template search consists of finding discrete prompts, which are actual text strings , or continuous prompts, where the prompt is described directly in the embedding space of the underlying LM.

Several methods make an automated discovery of discrete prompts. Prompt mining is a mining-based method to automatically find templates given a set of training inputs **x** and **y**. This method scrapes a large text corpus for strings containing **x** and **y** and finds middle words or dependency paths for prompt creation. Paraphrasing-based approaches take in an existing prompt, which was mined or manually constructed, and paraphrase them into a set of candidate prompts. To decide on a single prompt, the model's performance is usually watched on the specific task, and the best performing prompt is used. Gradient-based search applies a gradient-based search over actual tokens to find short sequences that can trigger the LM to generate the desired prediction. Some works treat the generation of prompts as a text generation task and use LM to develop the whole prompt for the input **x** and filling **y**. Prompt scoring builds upon the idea of manual prompt creation. After prompts are hand-crafted, they are scored with the LM to choose the prompt with the highest probability.

Prompts do not need to be for human consumption as they are used to perform a task effectively. Rather than being natural language prompts, continuous prompts perform prompting in the embedding space of the model. There are several approaches for creating effective continuous prompts. Prefix tuning is a method that prepends a sequence of continuous vectors to the input while keeping the model parameters frozen. Optimization is done according to the (4.1) where the log-likelihood objective is maximized while having a trainable prefix matrix M_ϕ and a fixed pre-trained LM parameterized by θ.

$$\max_{\phi} \log P(\boldsymbol{y} \mid \boldsymbol{x}; \theta; \phi) = \max_{\phi} \sum_{y_i} \log P\left(y_i \mid h_{<i}; \theta; \phi\right) \tag{4.1}$$

Some methods initialize the search for a continuous prompt using a prompt that has already been created manually or with discrete prompt search methods. Initialising with discrete prompts can be a good starting point for practical prompt tuning. Hard-soft prompt hybrid tuning does not use purely learnable prompts. In this method, tunable embeddings are inserted into a hard prompt template.

Answer engineering, unlike prompt engineering, aims to search for an answer space $\mathcal{Z}$ which will be mapped to the actual output $\mathcal{Y}$ resulting in an effective predictive model. Two dimensions must be considered when performing answer engineering: answer shape and answer design method.

When choosing an answer shape, there are three choices:

- Tokens: One of the tokens in the LM's vocabulary
- Span: A multi-token span
- Sentence: A document or a sentence

Token or span answer shapes are usually used in the classification task, while the sentence shape is used in generation tasks.

Answer design methods deal with designing an appropriate answer space $\mathcal{Z}$ and its mapping to $\mathcal{Y}$. One of the methods of answer design is manual creation. In manual creation, the space of potential answers $\mathcal{Z}$ and its mapping to $\mathcal{Y}$ are crafted by a benchmark designer. The space of potential answers can be constrained and unconstrained. Unconstrained answer space consists of all tokens of the LM vocabulary. In these cases, a generated answer $\mathbf{z}$ is the same as the true output $\mathbf{y}$. In constrained answer space, the range of possible outputs is limited to specific tokens. Constrained answer space is used, for example, for classification or entity recognition, whereas unconstrained answer space is used for generation tasks. Manual design can be quite difficult, so similar to prompt engineering, several automatic procedures are introduced. Answer paraphrasing starts with an initial answer space $\mathcal{Z}'$ and paraphrasing broadens it to include more words. The created answer space is used collectively when filling the prompt, or just the best performing answer words are used. Prune-then-search is a similar method to answer paraphrasing, where initially, a pruned answer space of several plausible answers $\mathcal{Z}'$ is generated. Then a specific algorithm further searches over the pruned space to select a final set of answers. There is also a possibility of continuous answer search, which is explored only by a few works. One approach is to assign a virtual class token to each class label and optimize the token embedding for each class together with prompt token embedding.

When training the LM with or without prompts, five approaches differ in whether LM parameters are tuned, do prompts add new parameters to the total sum, and if they are tuned. The five approaches are shown in Table 4.2.

Table 4.2: Different tuning strategies

Strategy	LM Params	Prompt Params	
		Additional	Tuned
Promptless Fine-tuning	Tuned	–	–
Tuning-free Prompting	Frozen	×	×
Fixed-LM Prompt Tuning	Frozen	✓	✓
Fixed-prompt LM Tuning	Tuned	×	×
Prompt+LM Fine-tuning	Tuned	✓	✓

The area of prompt-based learning is extensive and novel, with numerous works dealing with different task formations and goals. Schick and Schütze (2020) introduced a PET model, which is created after two subsequent procedures. First, several prompts are designed for the in-domain task, and a language model is trained over the prompts with only few examples. After training, the model is used in a sampling regime to create labels for the unlabelled examples. The final model is trained over the created dataset to achieve better performance than the model trained only on a few labeled examples. Gao et al. (2020b) auto-generated prompt templates with a T5 model. They used generated demonstrations and demonstrations like in the GPT3 model, gaining better performance than standard few shot finetuning. Perez et al. (2021) proved that hyperparameter and prompt search in a true few-shot setting are quite hard and usually not better than just the random selection. Zhong et al. (2021a) introduced a meta tuning step where they collected several datasets and changed their sentence-label setup into a question prompt setup with a yes/no filling of the prompt. They finetuned the T5 model on the datasets of similar tasks and did the zero-shot evaluation on a downstream specific task. They found that meta-tuning can help the final performance of the model. Gao et al. (2020a) automatically created prompts by inserting a token or token sequence into a masked prompt with a T5 model. Li and Liang (2021) introduced prefix tuning where they added tunable vectors at the beginning of the prompt and kept the LM parameters fixed. By tuning only 0.1% of the parameters, they achieved better or similar performance on three NLG tasks. Lester et al. (2021a) used all sizes of the T5 model with prefix tuning. Using only 20-100 tokens long prefixes with just a fraction of tunable parameters, they got a boost in performance in the few-shot setting compared to the GPT3 model and full model tuning of the T5 XXL model. Zhong et al. (2021b) used the prompts from the LAMA (Kassner et al., 2021) benchmark for the initialization of the continuous prompts and a BERT base model for the prediction of relations. Logan et al. (2021) experimented with prompt-based tuning, finding that null prompt and continuous prompt tuning perform

similarly in a few-shot setting, but manually created prompts perform the best. They also have proven that prompt only finetuning performed far worse than other setups. Lester et al. (2021b) used a T5 model pretrained for another 100K steps with an LM objective to provide an extensive overview of the influence of hyperparameters in prompt-based learning. They varied prompt length, continuous prompt initialization, and training all of the parameters or freezing the model. Prompt tuning has shown to be a good tool for transferring the frozen model to several different tasks.

5. Personality

5.1. Personality tests

While there is no generally agreed-upon definition of personality, most theories focus on motivation and psychological interactions with the environment one is surrounded by. Some theories define personality through traits that can predict user behavior. On the other hand, behavioral-based approaches define personality from exterior learning and habits. Psychologists have made several different tests for detecting a user's personality. Some of those tests are the Big Five Inventory, Minnesota Multiphasic Personality Inventory (MMPI-2), Rorschach Inkblot test, Neurotic Personality Questionnaire KON-2006, Eysenck's Personality Questionnaire (EPQ-R), Myers–Briggs Type Indicator and other. We will focus on the two most common personality tests, Myers–Briggs Type Indicator and Big Five Inventory.

Big Five Inventory represents personality as five distinctive traits. The first of five traits is openness. Openness to experience is a general appreciation for art, emotion, adventure, unusual ideas, imagination, curiosity, and experience. People with high openness tend to be more creative and aware of their feelings. They are likely to hold unconventional beliefs and seek euphoric experiences. Those with low openness seek to gain fulfillment through perseverance and are described as pragmatic, data-driven, close-minded, and dogmatic. Conscientiousness displays self-discipline and strive for achievement against measures or outside expectations. High conscientiousness is often perceived as being stubborn and focused. Low conscientiousness in an individual is associated with flexibility and spontaneity. Extraversion encompasses the breadth of activities, surgency from external activity/situations, and energy creation from external means. Extraverts enjoy interacting with people and are perceived as full of energy. On the other hand, introverts have lower social engagement and energy levels than extraverts. They are often quiet, low-key, deliberate, and less involved in the social world. Agreeableness reflects individual differences in general concern for social harmony. Agreeable individuals usually get along with others and are considerate, kind, generous, trustworthy, helpful, and willing to compromise. Disagreeable individuals place self-interest above getting along with others. They are unconcerned with others and are less likely to extend themselves to others. Low agreeableness personalities are often competitive

and challenging people who can be seen as argumentative or untrustworthy. Neuroticism tends to experience negative emotions, such as anger, anxiety, or depression. It is sometimes called emotional instability or is reversed and referred to as emotional stability. Figure 5.1 represents all five traits.

Figure 5.1: Big Five Inventory (CareerFitter)

Myers–Briggs Type Indicator (MBTI) is an introspective self-report questionnaire indicating differing psychological preferences in how people perceive the world and make decisions. The test describes a person's personality in four opposing traits: introversion or extraversion, sensing or intuition, thinking or feeling, and judging or perceiving. Extraverted individuals tend to act, then reflect, then act further. Conversely, those who prefer introversion expend their energy through action, so introverts need quiet time alone to rebuild their energy. Sensing and intuition are the information-gathering functions. People who prefer sensing are more likely to trust information in the present, tangible, and concrete. On the other hand, those who prefer intuition tend to trust the information that is less dependent on the senses. Those individuals are more interested in future possibilities. Thinking and feeling are the decision-making functions. Those who prefer thinking tend to decide things from a more detached standpoint, measuring decisions by logic, causality, and according to a set of rules. Feeling characteristic is more associated with empathy for the situation and weighing the needs of people involved to achieve the best harmony and consensus. Judging or perceiving traits are associated with people's preference for using either the judging function (thinking or feeling) or their perceiving function (sensing or intuition) when relating to the outside world. Depending on the affiliation with one of the opposing four traits, there are sixteen different personalities which are represented by four letters and are shown in Figure 5.2

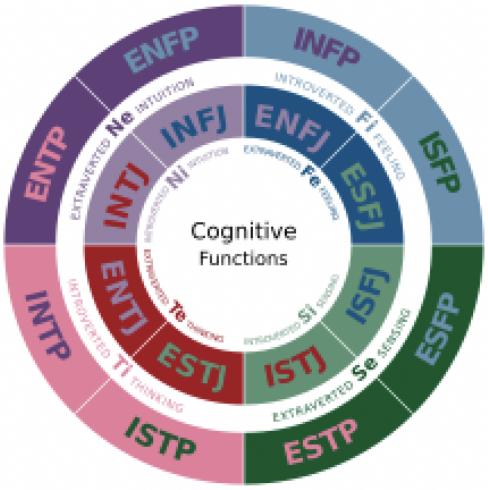

Figure 5.2: Myers–Briggs Type Indicator (MBTI) (Aeromagazine)

5.2. Personality assessment in NLP

Research attempting to connect language and personality has emerged very early. Pennebaker and King (2000) have tried to establish a link between personality and language use. Most research focused on automatic personality prediction from essays (Argamon et al., 2005), emails (Oberlander and Gill, 2006), conversations extracted from recorders (Mehl et al., 2001), Twitter (Quercia et al., 2011), or blogs (Iacobelli et al., 2011). The first use of large, user-generated content from social media was the MyPersonality dataset (Kosinski et al., 2015), with over 7.5 million Facebook user profiles labeled with Big Five types. Schwartz et al. (2013) used the MyPersonality database in the first large-scale personality prediction study based on text messages. Over 15.4 million statuses were collected. The study found that open-vocabulary approaches yield the best results in personality prediction. The growing interest in personality prediction resulted in two tasks (Pardo et al., 2015; Celli et al., 2013), which relied on benchmark datasets labeled with Big Five types. The results of these tasks have shown that n-gram-based models consistently yield good performance and that personality prediction is a hard task. Plank and Hovy (2015) made use of the general popularity of MBTI among the public and collected a dataset of over 1.2 million status updates on Twitter, and leveraged users' personality types. PANDORA (Gjurković et al., 2020) is one of the latest datasets from Reddit that continues on work of (Gjurković and Šnajder, 2018). It contains Big Five and MBTI types for over 10 thousand users and 17.5 million comments. Lately, most personality prediction models are based on deep learning (Majumder et al., 2017; Feizi Derakhshi et al., 2021; Rissola et al., 2019) as the area has shown great results on classification tasks. This work extends this research by considering the performance of models learned with prompt-based learning.

5.3. PANDORA

PANDORA (Persona and Demographics of Reddit Authors) (Gjurković et al., 2020) is the first dataset from Reddit labeled with personality and demographic data. PANDORA comprises over 17M comments written by more than 10k Reddit users, labeled with Big 5, MBTI, and Enneagram alongside age, gender, location, and language. The authors used the MBTI9k dataset (Gjurković and Šnajder, 2018) as a starting point to compile the dataset. In MBTI9k, authors relied on flairs to extract the MBTI labels. Flairs are short descriptions used for introduction in the subreddits, and on MBTI-related subreddits, they usually report the MBTI test results of the person. In total, 9054 users reported their MBTI type, and the PANDORA authors found another 793 additional Enneagram types. The distribution of the MBTI traits across users is shown in Table 6.2.

Table 5.1: Distribution of MBTI traits

MBTI Dimension	Users	MBTI Dimension	Users
Introverted	7134	Extraverted	1920
Intuitive	8024	Sensing	1030
Thinking	5837	Feeling	3217
Perceiving	5302	Judging	3752

Obtaining Big 5 labels turned out to be more challenging. Unlike other mentioned tests, the Big 5 test results in a score for each of the five traits, which is usually reported in different formats and the users reported them in the comments. The authors first retrieved candidate comments containing the three most likely traits to be spelled correctly (agreeableness, openness, and extraversion). For each comment, they retrieved the related post and determined the test version used according to the link in the post. Next, they automatically extracted the scores from commments and used manual verification if needed. This resulted in Big 5 scores for 1027 users. Left out from this procedure were the comments for which the test is unknown, as they were replying to posts without a link to the test. To detect tests for these comments, the authors trained a character n-gram model over the known posts, which was used to detect tests for unknown posts. This resulted in Big 5 scores for additional 600 users. The mean Big 5 scores are shown in Table 5.2 represented as percentile scores with a minimum of 0 and a maximum of 100. Mean for age is also shown, which was manually extracted together with locations from user flairs on different subreddits.

Table 5.2: Mean values for Big 5 percentile scores, age and number of comments

Big 5 Trait	All	Females	Males
Openness	62.5	62.9	64.3
Conscientiousness	40.2	43.3	41.6
Extraversion	37.4	39.7	37.6
Agreeableness	42.4	44.1	38.9
Neuroticism	49.8	51.6	46.9
Age	25.7	26.7	25.6
Comments	1819	2004	3055

6. Experiments

6.1. Preparation of datasets

The initial PANDORA dataset consists of 17.5 million comments. As the primary goal of this work is to provide an extensive comparison and performances of models trained in a classic setting, with hand-made prompts and with continuous prompts on a personality prediction task, the initial number of comments has shown to be too high to carry out effective experiments. The first reason is that most of the comments will not have indicative features that will provide information about the user's personality. Having comments that do not have essential information can cause the model to miss the goal of the task and not provide good results in personality prediction from a single comment. The second reason is that training base sizes of models over 17.5 million comments requires great time resources and limits the variety of experiments that can be carried out. For these reasons, I decided to design smaller datasets that will be used in later experiments.

According to the MBTI profiles, I created four smaller downstream datasets for each of the opposite traits. When designing these datasets as a first step, I only left users' comments with an entire MBTI personality present in the PANDORA dataset. This left me with 15.5 million comments from 9067 users. As we will predict the MBTI traits from a single comment, I paired the user's personality traits to all of its comments. After manual inspection of the comments, I found that numerous comments contain generic information from moderators, have a lot of hyperlinks, or do not contain natural language. To alleviate these issues, I filtered comments that contain any of the text patterns shown in Table 6.1. When dealing with comments that are not natural english language, I filtered comments that do not have at least one english stopword present. After filtering these unuseful comments, I also filtered those comments that do not have a minimum of 5 words or those that are longer than 256 words. After described filtering procedures, I was left with 12.1 million comments. The distribution of MBTI personality traits among all of the comments has changed compared to the initial PANDORA dataset and is shown in Table 6.2.

Table 6.1: Text patterns

Patterns
https://
http://
Thank you for your submission!
This comment or submission has been removed
All replies to this post must be a maximum
Here you can write whatever!
^^^^^^^

Table 6.2: Distribution of MBTI traits in comments

MBTI Trait	Num comments	%	MBTI Trait	Num comments	%
Introverted	9.6M	79.3	Extraverted	2.5M	20.7
Intuitive	10.5M	86.8	Sensing	1.6M	13.2
Thinking	8.9M	73.6	Feeling	3.2M	26.4
Perceiving	7M	57.8	Judging	5.1M	42.2

I decided on a specific training and testing regime to create four smaller datasets for the opposite traits. For every pair of opposite traits, I split all of the comments into a training split with two-thirds and a testing split with one-third of all the comments. I trained the pretrained language model over the train split and tested it over the test split. After getting the performance indicated with the macro F1 value, I extracted the model's confidence score for labeling that comment with a positive trait of the opposite trait pair. I chose that positive traits in the opposite trait pairs are going to be introverted, intuitive, thinking, and perceiving. I did this three times for every pair of the opposing traits, choosing a different third of data for testing. After this regime, all comments had four scores corresponding to the model's confidence score for the positive trait in trait pairs.

For training and inference, I used a TinyBERT (L=4, H=312, A=12) model which has 14.5M parameters. I chose a smaller model because training a larger model over millions of comments three times for each of the four pairs takes too much time resources. The train split was additionally split in a training split with 80% and a validation split with 20% of comments. The model was trained for ten epochs on the training part with early stopping according to validation F1 score with the patience of two epochs and best model saving. I used a batch size of 16 examples and an AdamW optimizer (Loshchilov and Hutter, 2017) with default parameters. To determine the best learning rate for the model, I carried out initial experiments on the introverted and extraverted pair using a learning rate of 5e-5, 2e-5,

1e-5, and 5e-6. According to the results on validation splits, I decided to use the learning rate with value 2e-5. I used a linear learning rate scheduler and lowered the comments before feeding them to the model. The implementation was done in Pytorch (Paszke et al., 2019), and all of the code for this regime and later experiments is available on Github.[1]

As the first results were quite bad because of a significant imbalance of opposing traits in the dataset, I decided to balance the dataset. I randomly sampled 1.5 million comments for every opposing trait, resulting in 4 distinct datasets of 3 million comments for every pair of opposite traits. This resulted in much better results which are displayed in Table 6.3. To show the macro F1 score, I averaged the three scores obtained during testing on different splits. The majority of comments will not have indicative features of a specific trait, so there was a significant number of comments that had wrongly predicted traits.

Table 6.3: Results of TinyBERT training

Traits	Acc
Extraverted/Introverted	58.18
Sensing/Intuitive	60.08
Thinking/Feeling	59.92
Judging/Perceiving	56.25

Using the confidence scores extracted during testing, I developed four datasets for each of the opposite traits. If the comment has indicative features of a specific trait, it will output a relatively higher or lower score for a positive trait than a non-confident score of 50%. To leave only the comments that most likely have indicative features, I decided on the thresholds of confidence scores represented in Table 6.4. I expressed the model's confidence score for a negative trait as 100% minus the confidence of a positive trait. After filtering, four final datasets had 300 thousand comments with a moderately balanced distribution of opposite traits which is shown in Table 6.4. Created datasets were used in later experiments in developing models for personality prediction.

[1] https://github.com/rcala1/PromptMBTI

Table 6.4: Thresholds for filtering

Trait	Num (%)	Threshold (%)	Trait	Num (%)	Threshold (%)
Extraverted	53.12	74.9	Introverted	46.88	72.6
Sensing	58.04	94.5	Intuitive	41.96	84.8
Feeling	48.86	76.3	Thinking	51.14	76.4
Judging	51.86	68.6	Perceiving	48.14	68.0

6.2. Classic setting

In a classic setting, I trained the language models in a more traditional way of fine-tuning a pretrained model for a classification task. I treated the prediction of opposite traits as a binary classification task. After being fed with the comment, the model was expected to output a probability of the comment showing a negative or a positive trait in the pair.

I used the BERT model and the GPT2 model. BERT model was modified with a linear layer over the pooled output, which is essentially the last hidden state of the [CLS] token further transformed with a linear layer and a tanh function. On the other hand, GPT2 model does not have a specific token whose hidden state is used for classification. GPT2 model is an autoregressive model with left-to-right decoding nature, so I used the last token's hidden state and added a linear layer for classification. The two models were trained in a full data setting, where all comments in created datasets were used and in a few-shot setting on just a few examples. Both regimes were tested on the same test split.

In the full data setting, I trained both models on four created datasets. The datasets were split into a train, validation, and test split with 70%, 10%, and 20% of data, respectively. Both models were trained for six epochs with early stopping with the patience of two epochs. I used an AdamW optimizer with default parameters and linear scheduling. To provide a fair comparison with later prompt-based learning, and because of limited resources, I used a batch size of 2 examples and lowercased the comments before feeding them to the model. I chose the base size for the BERT model (L=12, H=768, A=12) with 110 million parameters and a small size for the GPT2 model (L=12, H=768, A=12) with 117 million parameters. These model sizes were used later in prompt-based learning, as well. Each model has a different performance depending on the chosen learning rate. I carried out a hyperparameter search for the learning rate. Decision on the value of learning rate depended on the performance of the two models on the validation set for the introverted/extroverted pair. I chose to try values 5e-5, 2e-5, 1e-5, and 5e-6. In the end, learning rate of 1e-5 was used for the BERT model and 2e-5 for the GPT2 model. The results are displayed in Table 6.5.

Table 6.5: Full data setting F1 scores

Traits	BERT	GPT2
Extravert/Introvert	75.84	74.27
Sensing/Intuitive	84.76	83.19
Thinking/Feeling	77.94	76.80
Judging/Perceiving	70.44	68.82

The GPT2 model performed relatively worse than the BERT model, with approximately a 1% lower F1 score for all four traits. We can argue that the lower score is because BERT uses the output of a [CLS] token for classification. The output of the [CLS] token was used in the next sentence prediction task (NSP) in pretraining, resulting in better hidden state representation and better finetuning for a downstream classification task. The last token hidden state of the GPT2 model always comes from a different word that was last fed to the model, so it relatively changes and does not have a whole sentence context well embedded as the [CLS] token. This could be alleviated by shorter pretraining of the GPT2 model on a similar task as NSP on a larger text corpus, but I leave this idea for future research. For the BERT model, the best result was obtained for the sensing/intuitive pair with a high F1 score of 84.76%. The lowest score was obtained for the judging/perceiving pair with an F1 score of 70,44%.

There are different attitudes on how large can be the training set to stay true to the few-shot setting. I used 48 examples for the train split, 16 examples for the validation split, and the same test split as in full data setting to compare the two methods. The models were trained for 100 epochs with early stopping with patience of 5 epochs. The remaining regime and parameters for the BERT and GPT2 model were the same as in the full data setting. As the few-shot setting has high variance because of the small training set, I repeated the experiments three times over different random seeds and averaged the macro F1 scores obtained on the test split. The results are represented in Table 6.6.

Table 6.6: Few-shot setting F1 scores

Traits	BERT	GPT2
Extravert/Introvert	56.85	48.65
Sensing/Intuitive	59.90	45.99
Thinking/Feeling	61.97	43.92
Judging/Perceiving	50.52	46.54

The GPT2 model performed quite worse than the BERT model. The reason still lies in the fact that the BERT model uses the hidden state of the [CLS] token. In the few-shot setting, GPT2 did not have enough examples as in the full data setting to finetune the parameters to output a good classification score from using the last token hidden state. This results in a significant performance gap. The best result of 61.97% F1 score was obtained for the thinking/feeling pair, while again, for the judging/perceiving pair, I got the worst F1 score of 50.52%. F1 scores of the few-shot setting lag quite behind the scores of the full data setting. Predicting a personality trait from the comments is quite a difficult task that is demanding even for humans. Because of the task's difficulty and newly added linear layers, the model requires much more examples to learn the task of personality prediction.

6.3. Fixed-prompt LM Tuning

Fixed-prompt LM Tuning is an approach in prompt-based learning where a prompt is kept fixed and the model parameters are tuned for a downstream task. The prompt is usually hand-crafted or automatically searched by different methods. The created prompt can be a prefix prompt or a cloze prompt depending on a specific task and the underlying language model used.

For the task of personality prediction, I hand-crafted five prompts. The structure for every prompt was chosen by relating the personality prediction task to other tasks and hand-crafted prompts that have been shown to behave well (Liu et al., 2021). Manually created prompts are shown in Table 6.7. The model's goal during training and inference is to fill the prompt with one personality trait from the opposite trait pair that relates the best to the prompt and personality trait in the comment.

Table 6.7: Hand-crafted prompt examples

Prompts
1. prompt - I don't think Goku nor his family took any offense to that, xdfp extroverted/introverted? introverted
2. prompt - Text : I don't think Goku nor his family took any offense to that, xdfp MBTI : introverted
3. prompt - I don't think Goku nor his family took any offense to that, xdfp Based on the previous text, the person is of MBTI personality trait introverted
4. prompt - I don't think Goku nor his family took any offense to that, xdfp Question : What is the MBTI personality trait of the person in the previous text ? Answer : introverted
5. prompt (null prompt) - I don't think Goku nor his family took any offense to that, xdfp introverted

The BERT model was pretrained on a masked language modeling task. To stay true to the pretraining setup, I masked the true trait from the prompt and tasked the model to predict the masked part with the correct personality trait. The BERT tokenizer uses the Wordpiece vocabulary, which tokenizes one word into multiple tokens. Because of this, I masked the true trait from the prompt by putting the [MASK] token of the pretraining task instead of every token relating to the personality trait. At first, I got a perfect F1 score of the model on the validation split, which should not be the case. With further introspection, I found that different traits have a different number of tokens represented. This results in the model accurately predicting the masked personality trait by attending to the number of [MASK] tokens in the not filled prompt. I avoided this issue by adding [UNK] tokens next to traits that

get tokenized in fewer tokens than their opposite trait. The BERT model also had problems representing "extraverted" in tokens, so I used the word "extroverted" as a prompt filling for the BERT and GPT2 models for the introverted/extraverted pair.

The GPT2 model is an autoregressive model which was pretrained by predicting the next word conditioned on the previous words in the text. To stay true to the pretraining task, I fed the prompts to the model without a personality trait and tasked the model to fill them with the correct trait by conditioning on the remaining part of the prompt. First results were bad as I calculated the autoregressive loss over all of the words in prompt causing the model to not just learn the fillings, but the nature of the comments as well. To alleviate this issue, I modified the model to just calculate the loss over the prompt filling and not for the other words in the prompt, which gave me much better results. I applied this regime on the BERT model as well, calculating only the loss over the masked tokens.

For the full data setting, I split the downstream datasets into a train, validation, and test split with 80%, 10%, and 20% of examples, respectively. The test splits were the same as those in the previous classic training setting to provide a fair comparison later between classic model training and prompt-based training. I trained the models for six epochs with early stopping with the patience of two epochs. The learning rates used were the same as in classic model training. Optimizer was the AdamW optimizer with default parameters, and batch size used was two examples. The text was lowercased before it was fed to the models. Scheduling of the learning rate was done with a linear scheduler.

After the models were trained, I fed the prompts from the test split without a personality trait and tasked the models to predict them. As personality prediction is a classification task, I used a usual practice in prompt-based learning where the model is limited to output only the actual prompt fillings. For example, for the introverted/extraverted pair, the model can only output introverted or extroverted. The results for the full data training are represented in Table 6.8.

Table 6.8: Full data setting F1 scores

Traits	BERT				GPT2			
	I/E	I/S	T/F	J/P	I/E	I/S	T/F	J/P
1. prompt	75.62	84.59	**78.00**	**70.50**	75.40	84.05	77.62	69.46
2. prompt	75.81	84.37	77.96	70.35	75.37	84.42	77.77	69.64
3. prompt	75.75	84.32	77.91	70.01	75.68	84.31	77.72	69.51
4. prompt	**75.91**	84.35	77.90	70.35	75.49	84.00	77.62	69.49
5. prompt	75.61	**84.64**	77.93	70.40	75.62	84.03	77.63	69.07

The BERT model performed better than the GPT2 model on every prompt and trait pair. This can be because the BERT model is pretrained slightly better than the GPT2 model, and an MLM task suits the prompt classification of personality better than the autoregressive task. Comparing the performances between the prompts, we cannot see a significant difference in performance. The first prompt with the BERT model has, on average, the best results, but on the extraverted/introverted pair, it has a somewhat lower score.

Hand-made prompts are used in prompt-based learning the most in the few-shot setting. Training in a classic few-shot setting usually results in bad performance because a few examples are not enough to modify the added linear layers used for classification. Additionally, it is hard to learn useful information about the downstream task. To better extract the ingrained knowledge of the model, prompts has shown to be a better choice than the classic setting that guides the model in the right direction.

In the few-shot setting, I used the same five prompts as in the full data setting. I used 48 examples for the train split and 16 for the validation split. The examples used were the same as in the classic few-shot setting to provide a fair comparison later between the approaches. I used the same parameters and regimes as in the full data setting, but this time trained the models for 100 epochs with early stopping with the patience of 5 epochs. I repeated training and testing with different random seeds three times and averaged the F1 scores. The results obtained are shown in Table 6.9.

Table 6.9: Few data setting F1 scores

Traits	BERT				GPT2			
	I/E	I/S	T/F	J/P	I/E	I/S	T/F	J/P
1. prompt	**60.00**	60.67	61.92	51.36	59.88	59.49	57.95	53.01
2. prompt	58.62	**64.60**	63.51	51.97	56.41	64.58	**67.50**	49.46
3. prompt	56.80	57.46	53.14	**53.04**	56.13	54.56	64.82	44.63
4. prompt	51.05	60.97	60.69	46.69	54.37	58.14	55.30	51.30
5. prompt	56.40	62.45	58.88	48.98	53.86	64.44	64.91	49.47

Unlike in the full data setting, the two models performed relatively the same. The best performing prompt was the 2. prompt, which got the best F1 scores on the intuitive/sensing pair with the BERT model and on the thinking/feeling pair with the GPT2 model. Considering that in the full data setting best F1 score on the thinking/feeling pair was 78.00%, the F1 score in the few-shot setting of 67.50% is quite good because only 48 examples were used for training a challenging task as personality prediction. The 2. prompt is simple compared to the more complicated 3. and 4. prompts and has on average better performance. We can argue that simpler prompts in the few-shot setting perform better for personality prediction, and just a few indicators of the desired goal in the prompt can provide great performance.

The judging/perceiving pair follows the worst results in prediction as in the full data setting by being just better than random.

I tested the GPT2 model and the BERT model in the zero-shot setting, where I did not train the models and tasked them with filling the hand-made prompts. The results are shown in Table 6.10.

Table 6.10: Zero-shot F1 scores

Traits	BERT				GPT2			
	I/E	I/S	T/F	J/P	I/E	I/S	T/F	J/P
1. prompt	35.88	36.94	40.96	46.91	47.52	30.71	**61.87**	40.90
2. prompt	44.81	40.89	45.81	32.61	46.52	31.37	59.48	44.33
3. prompt	34.97	37.39	33.62	35.85	49.67	31.42	40.76	35.81
4. prompt	34.96	**54.47**	42.96	**48.08**	48.54	29.57	45.15	38.44
5. prompt	38.03	41.06	46.42	33.89	**50.40**	38.21	60.14	40.16

The results are bad and show that the models do not have broad ingrained knowledge to carry out personality prediction without any model training. This could be because the model sizes are significantly smaller than the models used in practice. Only on the thinking/feeling pair the GPT2 model got a good score, but overall the scores are like or worse than random.

6.4. Fixed-LM Prompt Tuning

Fixed-LM Prompt Tuning approaches the downstream task by freezing the model parameters and only tuning the parameters of the prompt. One tuning method is prefix tuning, where continuous embeddings are prepended to the embedded text, forming a continuous prompt. During training, these embeddings are learned according to the downstream task's goal. As the model is frozen, we can use different trained continuous prompts with the same model on numerous tasks, drastically saving on memory resources as continuous prompts usually have fewer than 0.1% parameters of the whole model.

For the task of personality prediction, I used prefix tuning. To create the continuous prompt, I prepended new special tokens to the text and only finetuned their embeddings while keeping the word embeddings and model frozen. Usually, the number of prepended embeddings varies between 20 to 100 newly added special tokens (Lester et al., 2021a). Because of the difficulty of the task of personality prediction, I decided to use a higher value of 100 prepended tokens. The performance of continuous prompts drastically depends on the initialization of newly introduced special token embeddings (Lester et al., 2021b). Random initialization often performs poorly, so the practice is to randomly initialize them with embeddings from the model's embedding layer, from the embeddings of a hand-made prompt, or with embeddings of true label words. In my experiments, I initialized the embeddings of newly introduced tokens by randomly sampling the word embeddings from the model's embedding layer.

The performance of continuous prompt training fluctuates and highly depends on the chosen hyperparameters. By using the learning rate of the models' in the classic setting and training with hand-made prompts, I got significantly worse results in the full data setting for prefix tuning. After a hyperparameter search, I found that tuning only special token embeddings works better with a higher learning rate. For both models, I settled on the learning rate of 7.5e-5.

In the full data setting, I split the comments in a train, validation, and test split with 80%, 10%, and 20% of comments, respectively. The splits are the same as in previous experiments to carry out a fair comparison between the approaches. The training lasted six epochs with early stopping with the patience of two epochs. I used a batch size of two and lowered the prompts before feeding them to the models. AdamW was used as an optimizer, and linear scheduling for the learning rate was incorporated. The results are shown in Table 6.11.

Table 6.11: Full data setting F1 scores

Traits	BERT	GPT2
Extravert/Introvert	70.89	60.51
Sensing/Intuitive	79.15	74.16
Thinking/Feeling	75.69	73.84
Judging/Perceiving	64.21	51.49

The BERT model performed significantly better than the GPT2 model. There could be a few reasons for the high difference between the performances. Continous prompts excessively depend on the hyperparameter choice, and the performance can vary greatly. The Wordpiece vocabulary of BERT is different than the BPE vocabulary of GPT2, so the initialization with the BERT vocabulary might have given the special tokens a better starting point for learning. The best performance was obtained on the sensing/intuitive part with an F1 score of 79.15%. Considering that the amount of trained parameters is less than 0.1% of the whole model, continuous prompt training has shown to be a good choice for high-performance memory efficient learning.

Continuous prompt training can be used in the few-shot setting. In this work, I trained continuous prompts on 48 examples in the train split and 16 in the validation split. I trained them for 100 epochs with early stopping with the patience of five epochs. The remaining parameters and regimes were the same as in the full data setting. The results are shown in Table 6.12

Table 6.12: Few data setting F1 scores

Traits	BERT	GPT2
Extravert/Introvert	54.79	47.78
Sensing/Intuitive	37.20	35.85
Thinking/Feeling	43.46	60.92
Judging/Perceiving	33.07	39.58

Prefix tuning in the few-shot setting got bad results with both models on the personality task. One reason for the very bad performance is that newly added embeddings do not have enough information to learn the personality prediction task with just a few examples. The second reason is that the models are too small to have enough ingrained knowledge to steer the learning of embeddings. Prompt-based learning models often have billions of parameters, so the models I used in my experiments have an order of magnitude less parameters.

6.5. Comparing the approaches

The setup and the models in the experiments were kept similar not just to develop effective models for personality prediction, but also to provide a comparison between the three approaches.

If we look at the best performances in the full data setting with the classic learning and hand-made prompts, we can see that there isn't a difference between the performances. When learning the model in the full data setting for personality prediction, it would be best to use the classic model setting as hand-made prompts can sometimes under-perform for one of the trait pairs. The performance of continuous prompts lacks behind, but overall results are quite good. With less than 0.1% parameters, continuous prompt learning managed to get close to full model training. Lester et al. (2021b) have shown that with a model of two orders of magnitude larger than the models used in this work, we can achieve the same results by tuning the model or only the prompt. If we use larger models, it would be best to use continuous prompt learning because we get the same performance without the need to save different finetuned large models.

In the few-shot setting, the best results of hand-made prompts outperform classic learning and continuous prompt learning. This follows the practices in the field, where hand-made prompts are a go-to method of learning large models in the few-shot setting. Classic few-shot learning is ineffective because few examples are insufficient to finetune the newly added linear layers for classification. Continous prompts are unstable and require accurate hyperparameter choice in the full data setting to perform well. The optimization becomes even harder in the few-shot setting, and few examples are not enough to properly learn the embeddings of special tokens. Larger models would help with the stability of continuous prompt learning and result in better performance. Still, the performance gain of large models is usually higher with the hand-made prompts. Highly efficient hand-made task prompts extract the information from large models very well and provide great performance with just little training.

The zero-shot setting has proven that models do not have broad ingrained knowledge of the personality prediction task to perform good classification without any training. This could be because the models are quite small compared to larger models usually used in practice and because of the difficulty of personality prediction. For the personality prediction task in the zero-shot setting, it would be best to use models at least two orders of magnitudes larger and to test even more hand-made prompts that are more complex or simple than those used in experiments here. Prompts used in this work can be a good starting point for the search of better ones in the zero-shot setting and the few-shot setting.

7. Conclusion

Personality is one of the main characteristics of an individual. Knowing the person's personality can predict behavioral preferences, psychological states, and how the environment affects them. This is of interest primarily for businesses because this gives them the information needed for targeted marketing, attracting customers, and increasing user retention.

Works dealing with personality prediction have shown that it is possible to develop highly efficient models to predict a person's personality from their text. The research has recently moved from machine learning to deep learning models. This work continues on the ideas of deep learning for personality prediction and investigates the effectiveness of prompt-based learning in personality prediction. We have shown that if we train models in the full data setting for personality prediction, it is best to use classic model training instead of prompt-based learning. As the used models get larger, from results obtained in this and previous works, it is better to use continuous prompts. Continous prompts eliminate the need to train and save large language models and require only for the prompt to be saved after training. Continous prompts have just a fraction of the parameters of the whole model, and their performance becomes the same as full model training if we use large language models. For the few-shot setting, we have shown that it is best to use hand-made prompts. Hand-made prompts steer the model better by giving it extra knowledge about the task and helping it learn the knowledge needed for prediction from just a few examples.

As the area of prompt-based learning is relatively new, there are numerous research questions that we can pursue in the future. Three approaches shown in this work can be tried with larger language models whose extra parameters will cause an increase in performance. Future work should automatically search or manually create more hand-made prompts that may behave better than the ones used in this work. Hand-made prompts in this work can act as a good starting point for future search.

BIBLIOGRAPHY

Aeromagazine. Mbti test. URL https://areomagazine.com/2021/03/09/should-you-trust-the-myers-briggs-personality-test/.

Shlomo Argamon, Dhawle S, Moshe Koppel, and James Pennebaker. Lexical predictors of personality type. 01 2005.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

CareerFitter. Big five personality test. URL https://www.careerfitter.com/career-tests/the-big-5-personality-test.

Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. Workshop on computational personality recognition: Shared task. In *ICWSM 2013*, 2013.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

Ali Reza Feizi Derakhshi, Mohammad Reza Feizi Derakhshi, Majid Ramezani, Narjes Nikzad Khasmakhi, Meysam Asgari-Chenaghlu, Taymaz Akan-R.Farshi, Mehrdad Khadivi, Elnaz Zafarni-Moattar, and Zoleikha Jahanbakhsh. The state-of-the-art in text-based automatic personality prediction, 10 2021.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners, 2020a. URL https://arxiv.org/abs/2012.15723.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners, 2020b. URL https://arxiv.org/abs/2012.15723.

Matej Gjurković, Mladen Karan, Iva Vukojević, Mihaela Bošnjak, and Jan Šnajder. Pandora talks: Personality and demographics on reddit, 2020. URL https://arxiv.org/abs/2004.04460.

Matej Gjurković and Jan Šnajder. Reddit: A gold mine for personality prediction. pages 87–97, 01 2018. doi: 10.18653/v1/W18-1112.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Francisco Iacobelli, Alastair Gill, Scott Nowson, and Jon Oberlander. Large scale personality classification of bloggers. In Sidney D'Mello, Arthur Graesser, Björn Schuller, and Jean-Claude Martin, editors, *Affective Computing and Intelligent Interaction*, Lecture Notes in Computer Science, pages 568–577. Springer-Verlag GmbH, 2011. ISBN 978-3-642-24570-1. doi: 10.1007/978-3-642-24571-8_71.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. Multilingual lama: Investigating knowledge in multilingual pretrained language models, 2021. URL https://arxiv.org/abs/2102.00894.

Michal Kosinski, Sandra Matz, Samuel Gosling, Vesselin Popov, and David Stillwell. Facebook as a research tool for the social sciences. *The American psychologist*, 70:543–556, 09 2015. doi: 10.1037/a0039210.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019. URL https://arxiv.org/abs/1909.11942.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021a. URL https://arxiv.org/abs/2104.08691.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021b. URL https://arxiv.org/abs/2104.08691.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021. URL https://arxiv.org/abs/2107.13586.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Robert L. Logan, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models, 2021. URL https://arxiv.org/abs/2106.13353.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL https://arxiv.org/abs/1711.05101.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79, 2017. doi: 10.1109/MIS.2017.23.

Matthias Mehl, James Pennebaker, D. Crow, James Dabbs, and John Price. The electronically activated recorder (ear): A device for sampling naturalistic daily activities and conversations. *Behavior Research Methods, Instruments, Computers*, 33:517–523, 11 2001. doi: 10.3758/BF03195410.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013a. URL https://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013b. URL https://arxiv.org/abs/1310.4546.

Jon Oberlander and Alastair J. Gill. Language with character: A stratified corpus comparison of individual differences in e-mail communication. *Discourse Processes*, 42(3):239–270, 2006. ISSN 0163-853X. doi: 10.1207/s15326950dp4203_1.

Francisco Manuel Rangel Pardo, Fabio Celli, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. Overview of the 3rd author profiling task at pan 2015. In *CLEF*, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep\-learning-library.pdf.

James Pennebaker and Laura King. Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, 77:1296–312, 01 2000. doi: 10.1037//0022-3514.77.6.1296.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models, 2021. URL https://arxiv.org/abs/2105.11447.

Barbara Plank and Dirk Hovy. Personality traits on Twitter—or—How to get 1,500 personality tests in a week. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 92–98, Lisboa, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-2913. URL https://aclanthology.org/W15-2913.

Daniele Quercia, Michal Kosinski, David Stillwell, and Jon Crowcroft. Our twitter profiles, our selves: Predicting personality with twitter. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 180–185, 2011. doi: 10.1109/PASSAT/SocialCom.2011.26.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019. URL https://arxiv.org/abs/1910.10683.

Esteban Andres Rissola, Seyed Ali Bahrainian, and Fabio Crestani. Personality recognition in conversations using capsule neural networks. In *IEEE/WIC/ACM International Conference on Web Intelligence*, WI '19, page 180–187, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369343. doi: 10.1145/3350546.3352516. URL https://doi.org/10.1145/3350546.3352516.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2020. URL https://arxiv.org/abs/2001.07676.

H. Schwartz, Johannes Eichstaedt, Margaret Kern, Lukasz Dziurzynski, Stephanie Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin Seligman, and Lyle Ungar. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8:e73791, 09 2013. doi: 10.1371/journal.pone.0073791.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Wikipedia contributors. Artificial intelligence — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=1092414501, 2022a. [Online; accessed 04-June-2022].

Wikipedia contributors. Big five personality traits — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Big_Five_personality_traits&oldid=1091508369, 2022b. [Online; accessed 04-June-2022].

Wikipedia contributors. F-score — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=F-score&oldid=1086969326, 2022c. [Online; accessed 04-June-2022].

Wikipedia contributors. Myers–briggs type indicator — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Myers%E2%80%93Briggs_Type_Indicator&oldid=1092337084, 2022d. [Online; accessed 04-June-2022].

Wikipedia contributors. Natural language processing — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=1092550276, 2022e. [Online; accessed 04-June-2022].

Wikipedia contributors. Personality — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Personality&oldid=1091468803, 2022f. [Online; accessed 04-June-2022].

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019. URL https://arxiv.org/abs/1906.08237.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections, 2021a. URL https://arxiv.org/abs/2104.04670.

Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [mask]: Learning vs. learning to recall, 2021b. URL https://arxiv.org/abs/2104.05240.

Personality prediction from text using prompt-based learning

Abstract

Personality is a characteristic feature that makes an individual unique in the population and is manifested as a set of opposing personality traits. As it is essential for recognizing and predicting human behavior, emerging research focuses on building models and approaches for personality prediction. Following the latest deep-learning methods, this work examines the use of prompt-based learning in personality prediction on the PANDORA dataset. With GPT2 and BERT models, this work explores prompt-based techniques in a zero-shot, few-shot, and full data setting. An extensive comparison is made between continuous and hand-made prompts and classic model training with different data availability. The results indicate that classic model training is not universal and that prompt-based approaches should be incorporated with less data and larger models.

Keywords: Personality, Prompt, GPT2, BERT, PANDORA

Predviđanje osobnosti iz teksta usmjeravanjem jezičnih modela

Sažetak

Osobnost je karakteristično svojstvo što čini pojedinca jedinstvenim u populaciji i manifestira se kao skup oprečnih crta ličnosti. Kako je ono ključno za prepoznavanje i predikciju ljudskog ponašanja, izranjajuće istraživanje se fokusira na izgradnju modela i pristupa za predikciju osobnosti. Prateći najnovije pristupe temeljene na dubokom učenju, ovaj rad proučava predikciju osobnosti na temelju usmjeravanja jezičnih modela pomoću predložaka i skupa podataka PANDORA. Pomoću GPT2 i BERT modela, ovaj rad proučava metode temeljene na predlošcima u okolinama bez podataka, s malo podataka i s potpunom raspoloživosti podacima. Detaljna usporedba je napravljena između kontinuiranih i ručno izrađenih predložaka te klasičnog treniranja modela u različitim okolinama dostupnosti podataka. Rezultati pokazuju da klasično treniranje modela nije univerzalno i da pristupi temeljeni na predlošcima trebaju biti ukomponirani u okolinama niske dostupnosti podataka ili u treniranju velikih modela.

Ključne riječi: Osobnost, Predložak, GPT2, BERT, PANDORA