ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

PROMPTING TECHNIQUES FOR NATURAL LANGUAGE GENERATION IN THE MEDICAL DOMAIN

CANDIDATE
Martina Rossini

SUPERVISOR
Prof. Paolo Torroni

CO-SUPERVISORS
Prof. Elena Cabrio
Dr. Serena Villata

Academic year 2021-2022

Session 3rd

Abstract

Generating automatic explanations for the predictions of machine learning models has long since been a major challenge in Artificial Intelligence, especially when considering sensible domains like healthcare. In this thesis, we approach the problem of generating a fluent, natural language justification for a medical diagnosis given all the information in the case description and the disease's symptomatology. We treat this problem as a data-to-text task and solve it using prompting techniques for natural language generation. In particular, we propose two architectural modifications to standard Prefix Tuning called Layer Dependency and Prefix Pooling; we evaluate their results, comparing with current state-of-the-art methods for the data-to-text task, on both a general-purpose benchmark (WebNLG) and on a dataset of clinical cases and relative explanations built as part of the ANTIDOTE project. Results show that layer dependency boosts the generation capabilities of our models when training on a limited computational budget, while Prefix Pooling is a valid dynamic prompting technique that achieves performances on par with the current state-of-the-art without requiring any additional information to be associated with the input. Finally, we note how, for our domain and in the context of the ANTIDOTE project, interpreting the explanation-generation task as data-to-text is a viable approach which produces high-quality justifications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Generating automatic explanations for the predictions of machine learning models has long since been a major challenge in Artificial Intelligence. In recent years the topic has become even more important, due to both the rise in research on neural black-box models and the increasing number of applications of such models in sensitive domains. One of these domains is healthcare, in which it is fundamental that the model is not only able to provide a diagnosis based on a clinical case, but also to explain the reasoning that led it to that conclusion. Some important contributions in this area [7, 48] point out how the provided explanations should be both convincing and easy to interpret, which lead to investigating the possibility of using natural-language, argument-based explanations.

This work was carried out as part of the European project ANTIDOTE[1] (funded by the CHIST-ERA 2019 call), which aims to provide a unified framework for learning to take decisions in the medical domain to provide justifications for said decisions. This is obviously a very broad and challenging task, which is tackled by a consortium of European universities and research centers. In particular, the French portion of the consortium - in collaboration with whom this thesis was carried out - is working on generating natural language argument-based explanations for educational purposes. In fact, medical residents are trained through standardized tests, where they are presented with a clinical case and have to decide which is the most likely diagnosis among a limited set of possibilities. They also have to provide an explanation for their decision. In order

[1]https://univ-cotedazur.eu/antidote

to produce a model that can automatize this process, at the current state of the project a full pipeline was designed that (i) detects the symptoms in the description of the clinical case, (ii) matches them with the symptoms that a medical knowledge base associates with the possible diagnosis and (iii) generates natural language explanations based on the previously extracted information. In particular, this work focuses on improving the last step, moving away from pattern-based explanations toward more complete, fluent, and discoursive justifications.

More precisely, since all the components necessary to explain a diagnosis based on patient symptoms had already been extracted, we decided to frame our problem as a data-to-text task. We structure the available information as triplets of three main types: `(patient, hasSymptom, symptom)`, `(patient suffersFrom, disease)`, and `(disease, hasAsSymptom, symptom)`. Our explanation is then obtained by verbalizing a triplet set containing all the patient's symptoms, the hypothesized diagnosis, and its associated symptomatology. The verbalization is produced using generative pre-trained language models like BART [16] and GPT-2 [37]. While designing this architecture, we had to solve some challenges and take into consideration some important requirements, like:

- The shortage of training data. Indeed, medical-domain datasets are small due to the significant labeling efforts required by expert personnel, but also due to privacy issues. In our case, we only had the 314 example medical questions annotated by Marro et al. [23]. We thus had to rely on augmenting the dataset with multiple reference verbalization per triplet set; we also devised smaller and easier inputs that basically represent individual portions of the whole explanation (i.e., only the patient symptoms or only the disease information).

- Lightweight fine-tuning of the language models. The small training set made it extremely challenging to approach the problem through standard fine-tuning while avoiding overfitting. We thus focused our attention on lightweight fine-tuning strategies, and in particular on prompting [19] and Prefix Tuning [18], where the main idea is to concatenate a small number of additional tunable parameters to the keys and values of the attention at each layer of our model while keeping the rest of the architecture frozen. This also allows for easy model sharing.

- The need for controllable generation. Indeed, in the medical domain, it is especially important to be able to steer the generated text toward the preferred topic, as well as to be able to ensure it has some important qualities. For instance, one might want the generated verbalization to be more technical when it has to be shown to a doctor or to a medical resident, while the same level of complexity is probably not desirable when the recipient is the patient.

Our work also focused on examining and comparing the performances of existing state-of-the-art prompting techniques [6, 18], as well as trying to improve them. Indeed, we first propose a slight modification to the standard Prefix Tuning [18] architecture which introduces an explicit prior telling the models that the prefix at layer i should depend on that at layer $i-1$. Extensive experiments are performed on the effectiveness of this dependency, showing that it tends to boost performances for smaller models and in particular when training with a limited computational budget. Next, we propose a new dynamic prompting technique, where the prefixes are input-dependent but, differently from Control Prefixes [6], do not require the presence of additional non-textual information acting as a guidance signal. Again, we conducted an array of tests comparing this technique with both Prefix Tuning and Control Prefixes, primarily showing that it is able to reach performances on par with them and, in some circumstances, also to outperform them. Notice that, since we proposed some architectural changes and advancements, we also employed a general-purpose data-to-text benchmark called WebNLG [10] in addition to the medical data collected in the ANTIDOTE project. This was done in order to render our models more directly comparable with those coming from the research community.
Summarizing, the main contributions of this thesis are the following:

1. We provide an analysis of the advantages and disadvantages of prompting techniques for natural language generation, with a particular focus on data-to-text tasks;

2. We propose two architectural modifications of the current state-of-the-art, aiming

to solve some weak points of existing techniques. For instance, the new Prefix Pooling approach addresses the impossibility of having a dynamic prefix that changes together with the input triplet, when no additional information is present. This would indeed be important, as within the same dataset, instance complexity can vary greatly.

3. We apply the aforementioned prompting techniques to the medical domain, showing that they are a viable approach to producing coherent explanations of a diagnosis in natural language. We also note that they overcome the limitations of the pattern-based justification employed by Marro et al. [23].

The remainder of this work is organized as follows; Chapter 2 presents in detail the context of the ANTIDOTE project, as part of which this thesis was carried out. Chapter 3 presents a broad overview of transformer-based pre-trained language models, with a specific focus on the three generative architectures which we exploit in the course of our experiments, that is, GPT-2 [37], BART [16] and T5 [38]. We also present in this chapter the medically-oriented versions of the aforementioned models, as well as give a primer on both lightweight fine-tuning strategies and the text generation task. Chapter 4 goes into more detail about the use of prompting techniques for natural language generation, formally introducing Prefix Tuning [18], Control Prefixes [6] and our two architectural modifications. Then, Chapter 5 presents the two datasets used during our experiments, WebNLG and USMLE-Symp. For each of them, we provide a set of statistics related to the input length in words and characters, as well as show some characterizing examples. Chapter 6 details the experimental studies we carried out: we first present the technological stack used to develop our models and track their performances during training, then explain the evaluation process, which varies together with the dataset, and - finally - report the various experiments we realized, together with their results and some qualitative examples of model performances. Chapter 7 offers a summary of the main results of our experiments and highlights any limitations or critical points of our work, while Chapter 8 draws some conclusions and presents possible improvements and future work. Finally, the appendices provide more details about the hyperparameters (Appendix A), illustrate multiple qualitative examples of the models' output for each dataset (Appendix B), and display some interesting visualizations to further analyze the performances (Appendix C).

Chapter 2

The ANTIDOTE Project

This work has been carried out in the context of the ANTIDOTE [1] project[1] with funding from the European coordinated research consortium on long-term ICT and ICT-based scientific challenges (CHIST-ERA). Indeed, this work was supported by the CHIST-ERA grant of the Call XAI 2019 of the ANR with the grant number Project-ANR-21-CHR4-0002.

ANTIDOTE - which stands for "*ArgumeNtaTIon-Driven explainable artificial intelligence fOr digiTal mEdicine*" - aims to provide a unified framework for both learning to take decisions in the clinical domain and providing an explanation of said decisions. Indeed, the correlation between the internal states of a deep neural network and its output is not well studied, which is a great limitation when applying these models to real-world scenarios. This is especially true in the medical domain, where the need for high-quality explanations is a critical factor. The project specifically focuses on argumentation-driven justifications because high-quality and human-provided explanations are often based on giving examples in support of the final decision, while pointing out the reasons for rejected alternatives - that is, they are based on argumentative mechanisms.

The ANTIDOTE project is carried out by a consortium of European universities and research centers under the coordination of professors Cabrio and Villata from the Côte d'Azur University in Sophia Antipolis, France. The French branch of the consortium, where the work for this thesis was realized, focuses its research efforts on the argument

[1]https://univ-cotedazur.eu/antidote

mining and generation part of the pipeline: in particular, their main contributions are two-fold. First, they implemented ACTA[2] [24, 26], a tool for automating the argumentative analysis of clinical trials, intending to support clinicians in the application of evidence-based medicine (EBM) - that is, the use of current best evidence and scientific information coming from systematic reviews to guide clinical decision making. The tool allows for automatic extraction of argumentative components from the abstract of a randomized control trial, as well as for the visualization of the relations (i.e., *support* or *attack*) between them. Moreover, ACTA can be used to extract elements of the PICO model[3] from the given abstract and to identify the effects of the trial on an outcome.

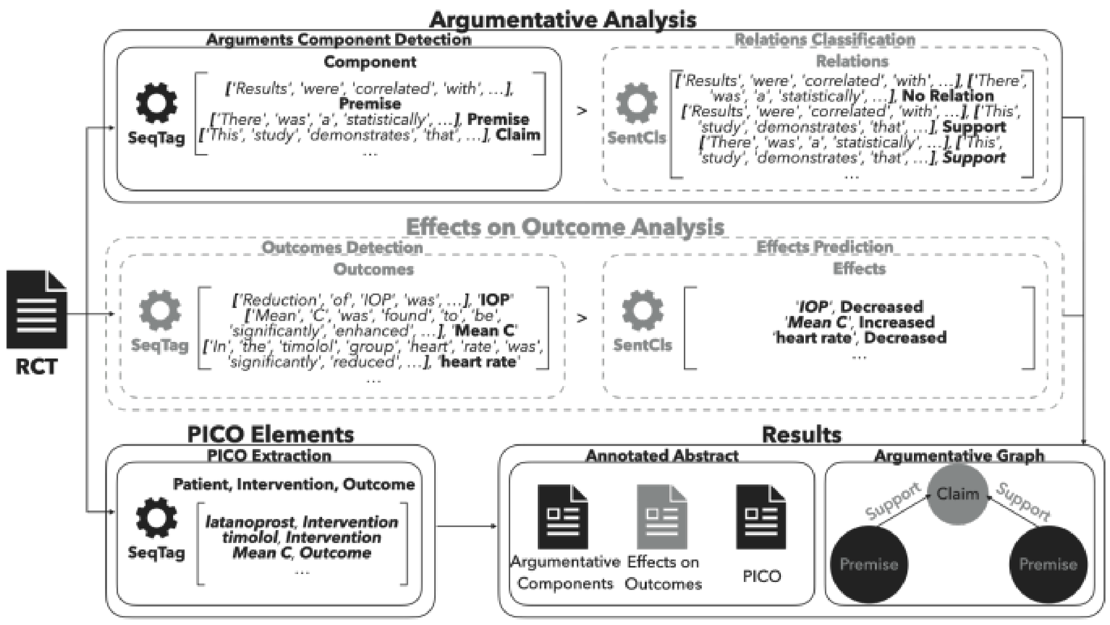

Argumentative Analysis
Arguments Component Detection
Component
SeqTag
['Results', 'were', 'correlated', 'with', ...], Premise
['There', 'was', 'a', 'statistically', ...], Premise
['This', 'study', 'demonstrates', 'that', ...], Claim
Relations Classification
Relations
SentCls
['Results', 'were', 'correlated', 'with', ...], ['There', 'was', 'a', 'statistically', ...], No Relation
['Results', 'were', 'correlated', 'with', ...], ['This', 'study', 'demonstrates', 'that', ...], Support
['There', 'was', 'a', 'statistically', ...], ['This', 'study', 'demonstrates', 'that', ...], Support
Effects on Outcome Analysis
Outcomes Detection
Outcomes
SeqTag
['Reduction', 'of', 'IOP', 'was', ...], 'IOP'
['Mean', 'C', 'was', 'found', 'to', 'be', 'significantly', 'enhanced', ...], 'Mean C'
['In', 'the', 'timolol', 'group', 'heart', 'rate', 'was', 'significantly', 'reduced', ...], 'heart rate'
Effects Prediction
Effects
SentCls
'IOP', Decreased
'Mean C', Increased
'heart rate', Decreased
RCT
PICO Elements
PICO Extraction
Patient, Intervention, Outcome
SeqTag
latanoprost, Intervention
timolol, Intervention
Mean C, Outcome
Results
Annotated Abstract
Argumentative Components
Effects on Outcomes
PICO
Argumentative Graph
Claim
Support
Support
Premise
Premise

Figure 2.1: Overview of the ACTA [24, 26] system for argumentative components extraction from the abstract of a clinical trial.

Second, they are focusing on generating natural language argument-based explanations to be used for educational purposes in the medical domain. Indeed, medical students are often trained through standardized tests where they are given a clinical case and asked to identify the most likely diagnosis from a pre-defined set of possibilities, while also explaining their reasoning. Figure 2.2 gives an overview of the proposed

[2]https://ns.inria.fr/acta/

[3]PICO [39] is a format to help clinicians formulate their clinical questions. The acronym stands for Population or Patient, Intervention, Comparison/Control, and Outcome.

pipeline for this system [23], which:

1. Detects symptoms and patient information from the clinical case

2. Matches the detected symptoms with those in a medical knowledge base to identify the disease(s) they are associated with. The reference knowledge base is the Human Phenotype Ontology (HPO) [15], a comprehensive Web Ontology Language (OWL) that systematically defines human phenotypes and also organizes them logically. It has broad coverage, currently containing more than 13k terms, and is thus an important resource for most - if not all - of today's computational disease models.

3. Generates pattern-based natural language explanations of two main types, depending on whether the interest is on justifying the model's prediction with positive examples or on explaining why a different alternative was rejected. A third template is used to enrich the explanations with additional arguments, for instance by drawing the reader's attention to other statistically important symptoms for the correct disease that are not explicitly mentioned in the clinical case. Table 2.1 shows the pattern used for all three types of argumentations.

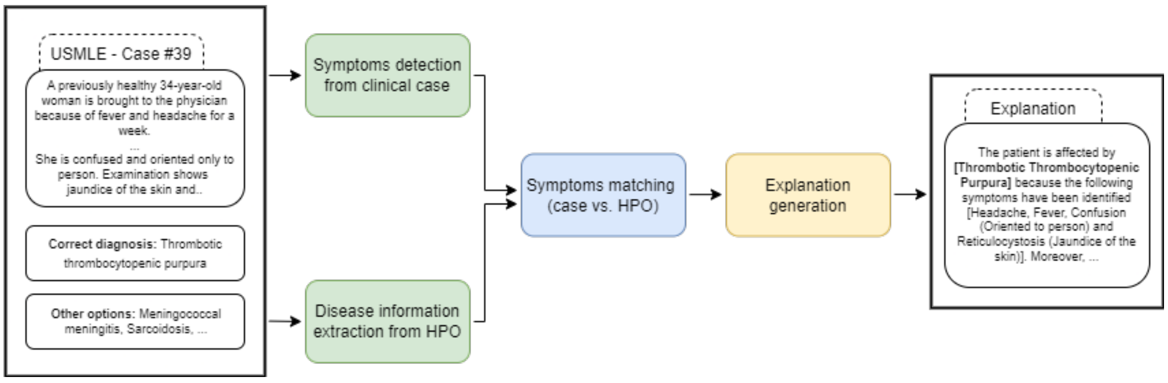

USMLE - Case #39
A previously healthy 34-year-old woman is brought to the physician because of fever and headache for a week.
...
She is confused and oriented only to person. Examination shows jaundice of the skin and..
Correct diagnosis: Thrombotic thrombocytopenic purpura
Other options: Meningococcal meningitis, Sarcoidosis, ...
Symptoms detection from clinical case
Disease information extraction from HPO
Symptoms matching (case vs. HPO)
Explanation generation
Explanation
The patient is affected by [Thrombotic Thrombocytopenic Purpura] because the following symptoms have been identified [Headache, Fever, Confusion (Oriented to person) and Reticulocystosis (Jaundice of the skin)]. Moreover, ...

Figure 2.2: Overview of the explanation generation pipeline

Our work focuses on improving the aforementioned natural language explanations, moving away from the single templates and towards more complex discursive justifications. In particular, this is achieved by feeding all the components we previously extracted from the clinical case (and on which the pattern-based explanation is based)

Table 2.1: Pattern-based post-hoc explanations

Pattern Type	**Template**
Why template	The patient is affected by [***correct diagnosis***] because the following symptoms have been identified [*list of matched symptoms from HPO*]. Moreover, [*list of obligatory symptoms*] are obligatory symptoms (always present, i.e., 100% of the cases) and [*list of frequent symptoms*] are very frequent symptoms (holding in 80% to 90% of the cases) for [***correct diagnosis***] and are present in the case description.
Why-not template	The diagnosis [*wrong diagnosis*] has to be discarded because the patient is not showing the symptoms [*list of missing symptoms*], which cannot be found in the case description. Moreover, [*list of obligatory symptoms*] are obligatory symptoms (always present, i.e., 100% of the cases), and [*list of frequent symptoms*] are very frequent symptoms (holding in 80-90% of the cases) for [*incorrect diagnosis*] and they are not present in the description.
Additional-info template	Furthermore, [*list of symptoms*] are also very frequent symptoms for [***correct diagnosis***] and could be present in the findings part of the clinical case.

to a pre-trained generative language model like GPT-2. As will be explained in more detail in Section 5.2, we organize all the information extracted from the clinical case in a set of triplets and we frame the problem as a data-to-text task.

Finally, note that while designing this solution we had to consider some key requirements: for instance, we know that labeled data in the medical domain is scarce due to both patient privacy issues and significant labeling efforts. We should thus be able to perform lightweight fine-tuning of our language model. We also need to have some control over the generation, to steer it towards a certain topic or to specify the amount of details to use. Indeed, in the medical domain, one might want to provide a more detailed and comprehensive explanation to the clinician and a less complex but equally useful explanation to the patient. More details about the implemented architectures are given in Chapter 4, while the medical dataset is described in Chapter 5.

Chapter 3

Background

In recent years, deep learning solutions have been widely adopted for countless Artificial Intelligence (AI) applications, ranging from image classification to image captioning and natural language generation. Neural approaches are able to perform representation learning, that is, they can learn low-dimensional continuous vectors (i.e., *distributed representations*) encoding both general-purpose and task-specific features of the underlying training data. In doing so, they remove the need for a heavy feature engineering pipeline, where researchers and domain experts use their knowledge to extract interesting features from raw data [19]. Despite their popularity, it has been pointed out [11] that these new models are prone to over-fitting when the amount of training data is not sufficient, becoming unable to achieve good generalization on unseen data.

A fully supervised training setting - where we train a task-specific model from scratch on the relative dataset - has long since played a central role in the Machine Learning landscape. However, obtaining high-quality human annotations is an expensive and time-consuming process, more so when the task requires specific domain knowledge; because of this, real-world datasets are often not big enough to effectively train a deep learning model in a fully-supervised setting [11]. Therefore, inspired by the human ability to use previously acquired knowledge for solving new problems, transfer learning was introduced as a way to effectively train big neural models on a limited amount of human-annotated data. This technique is usually composed of two separate training steps: (i) a pre-training phase, to capture knowledge from one (or more) source tasks, and (ii) a fine-tuning phase, which tries to transfer the previously acquired knowledge to new target tasks [11, 35].

The transfer learning paradigm is very influential and it has been used in Computer Vision (CV) and Natural Language Processing (NLP) alike. Choosing a good task to solve during the pre-training phase is particularly important for learning good representations; in principle, we want the problem to be challenging while still having an abundance of training data. We can identify the following three main categories of pre-training tasks [35]:

1. *Supervised pre-training*: based on a labeled dataset, we learn a function that is able to map inputs to outputs.

2. *Unsupervised pre-training*: starting from unlabeled data, we extract some patterns or intrinsic information from it.

3. *Self-supervised pre-training*: a blend of supervised and unsupervised learning where the data is originally unlabeled and the ground truth information is automatically produced starting from the examples themselves. The basic idea behind this paradigm in NLP is to try and predict a masked part of the input starting from the remaining parts.

Supervised pre-training tasks are especially popular in CV, resulting in a long series of convolutional architectures pre-trained on the big ImageNet [8] dataset for visual recognition and then fine-tuned on custom data, either for the same task (i.e., image classification) or for a different one. Indeed, the weights of an image classification network can also be used as initialization for models tackling different tasks like object detection, where obtaining huge labeled datasets would be too expensive; it has been shown that this speeds up convergence and improves generalization [35].

Most common NLP tasks have higher annotation costs with respect to the Computer Vision counterparts, which makes obtaining labeled datasets of the same size as ImageNet extremely hard. Since constructing a large-scale unlabeled corpus is much easier, research efforts focused on learning good representations from unlabeled data, that is, researchers moved towards unsupervised or self-supervised pre-training paradigms.

Pre-trained Word Embeddings: This first generation of pre-trained models for natural language processing marks an initial attempt to learn how words can be represented as dense vectors starting from unlabeled data. Many different approaches were developed, some of which do not rely on neural architectures at all. For instance, GloVe [31] obtains pre-trained word embeddings by computing global word-to-word co-occurrence statistics from a large unlabeled corpus. Other relevant techniques like Word2Vec [25], instead, use shallow neural architectures trained to reconstruct the linguistic contexts of words. All of these approaches are able to capture latent semantic and syntactic similarities among words; for example, we might find that `embed("China") - embed("Beijing")` ≈ `embed("Japan") - embed("Tokyo")`. Moreover, the embedding vectors usually demonstrate peculiar properties like compositionality, meaning that we might observe `embed("Germany") + embed("Capital")` to be close to `embed("Berlin")` in the embedding space [35]. Some probing tasks [40] also showed that these vectors can capture hierarchical information, like the fact that a dog is an animal or that a woman is a human being. However, they do not capture contextual information and thus have problems with polysemous words, whose sense changes depending on the context. Since these words are very common in the English language, a first-generation pre-trained model cannot capture, for instance, that the word *bank* has two completely different meanings in the sentences *"I want to open a bank account"* and *"It was lying on the river bank"* [11]. Moreover, while these methods do improve the performance of NLP architectures on downstream tasks, they are only ever useful as an initialization prior for the embedding layer, which implies that the remaining parts of the architecture still have to be learned from scratch [11].

Pre-trained Contextual Encoders: In order to have a token's representation that changes with the context, second-generation pre-trained language models were introduced. They feed word vectors to a neural encoder and use its hidden state as a dynamic (or *contextual*) embedding; Figure 3.1 illustrates a generic neural architecture for NLP which includes a contextual encoder. Contextual encoders are typically divided into two main categories, sequence models - like convolutional or recurrent networks - and non-sequence models (i.e., Transformer-like architectures [49]), which typically use

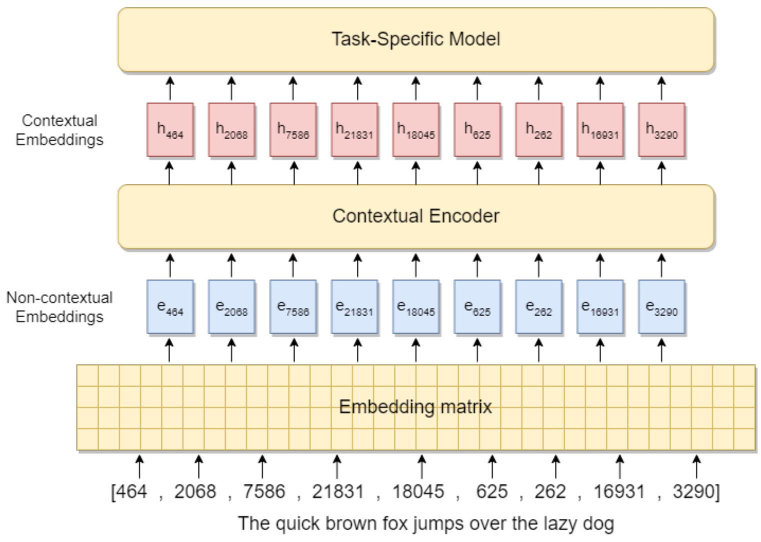

Task-Specific Model
Contextual Embeddings
h464 h2068 h7586 h21831 h18045 h625 h262 h16931 h3290
Contextual Encoder
Non-contextual Embeddings
e464 e2068 e7586 e21831 e18045 e625 e262 e16931 e3290
Embedding matrix
[464 , 2068 , 7586 , 21831 , 18045 , 625 , 262 , 16931 , 3290]
The quick brown fox jumps over the lazy dog

Figure 3.1: Neural architecture with contextual encoder. Image adapted from [35].

a fully-connected architecture to model the relation between words and then learn the actual linguistic structure of the sentence thanks to an attention mechanism. Sequence models can usually be trained with a smaller amount of data, but struggle to capture long-term dependencies between words. Probing tasks [19] showed that contextual word embeddings are able to capture both linguistic (i.e., syntax, grammar, and semantics for different word senses) and world knowledge present in the training data.

Modern Language Models (LMs) are usually pre-trained in a self-supervised setting to predict the probability that a word/sentence occurs in a span of text. Since we have abundant raw unlabeled data, we can use this phase to learn robust and general-purpose features for our language of choice. We typically consider the learned representation to be good if it captures both implicit linguistic rules and common sense information that might be hiding in the training corpus [35]. A Pre-Trained LM (PLM) is then fine-tuned in order to solve different downstream tasks. Note that, while this is currently the dominant solution in order to transfer the PLM's knowledge to a new task, it is not parameter efficient [11], which can be impractical and problematic in two main scenarios:

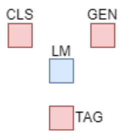
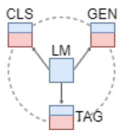
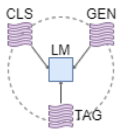
1. Multi-task systems, where we would need to store a separate copy of all the model's parameters for every downstream task we have to solve. Given the current

size of language models, this quickly becomes prohibitive in terms of memory requirements: consider, for instance, that GPT-2 [37] has 775M weights, while GPT-3 [4] parameter's number surpasses 100 billion.

2. Low-data situations, where we would have to fine-tune large-scale models with few labeled training examples, thus potentially incurring in over-fitting.

For this reason, different techniques for lightweight fine-tuning are being developed and research is shifting towards a "pre-train, prompt, predict" paradigm [19], whose main idea is to stop adapting the pre-trained LM to a downstream task. Instead, we make the downstream task more similar to the problem our model learned to solve during its pre-training phase. This is done through a prompt, which could either be textual or composed of learnable parameters; for instance, if we want to recognize the topic of the sentence *"The Italian team lost the match"*, we might feed it to the language model followed by the prompt *"The text is about _"* and ask it to fill the blank.

Table 3.1: Different paradigms in NLP. Table adapted from [19].

Paradigm	**Task relation**
Fully-supervised learning	
Pre-train, Fine-tune (Self-supervised pre-training)	
Pre-train, Prompt, Predict	

This new paradigm is particularly interesting in low-data scenarios because, by selecting an appropriate prompt, the pre-trained LM might be able to predict the desired output without needing any fine-tuning steps [19]. Table 3.1 shows how the process of training language models evolved during the years; note that the blue color means unsupervised training, the red one means supervised training, and the purple stands for prompting. Also notice that a connection `LM` $\rightarrow$ `Task` means that we are adapting the LM objective to a downstream task, while a connection `Task` $\rightarrow$ `LM` indicates that we are adapting the formulation of the downstream task to better align with the language model's objective. Finally, the dashed lines suggest that the parameters of the pre-trained LM can be shared across downstream tasks.

The remaining part of this chapter is organized as follows: Section 3.1 explains in more detail the architecture and the pre-training objective of common PLMs which are suitable for the objective of this thesis, that is, for text generation. Section 3.2 introduces lightweight fine-tuning strategies, highlighting the advantages and disadvantages of the various possibilities. Finally, Section 3.3 gives a primer on text generation, explaining how we can decode the output of a LM to obtain an actual sentence and detailing some common evaluation metrics for this complex task.

3.1 Pre-trained Language Models

Modern pre-trained language models are often variations of the Transformer architecture [49], a non-recurrent sequence-to-sequence model based on feed-forward layers and attention. The attention mechanism, in particular, is crucial as it is able to seamlessly draw connections between any two parts of the input sentence, thus resolving the problems with long-term dependencies typical of recurrent approaches. Moreover, we should note that Transformer-like architectures do not suffer from vanishing gradients and are more GPU-friendly than RNNs.

In psychology, attention can be loosely defined as the human ability to focus on certain stimuli, features, or events in the environment, while tuning out other details [29]; similarly, here we use attention to give more weight to certain parts of the input text. In particular, the attention function used in the Transformer is called *scaled dot-product attention* and takes as input queries and keys of dimension d_1 and values of dimension d_2. For every query vector q_i, we compute its dot product with the keys and divide the result by the square root of d_1 before feeding everything to a soft-max function; we obtain weights that are used to compute a weighted sum of the value vectors. Note that the scaling by $\sqrt{d_1}$ prevents large values in the dot products from dominating the attention weights and potentially pushing the soft-max function into regions of the space where it has small gradients [49]. Formally, given a set of queries $Q = \{q_1, \dots, q_n\}$, keys $K = \{k_1, \dots, k_m\}$ and values $V = \{v_1, \dots, v_m\}$ we can compute the matrix of outputs as shown in Equation 3.1.

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_1}}\Big)V \tag{3.1}$$

Instead of computing a single attention function with key, value, and query vectors of a dimension equal to the hidden size of the model (i.e., $d_{model} = 512$), the Transformer's authors actually devise a multi-head attention: they linearly project queries and keys into a smaller dimension d_1 for h different times and then do the same for the values but using dimension d_2. Each of these projected versions of Q, K, and V is used to compute the scaled dot-product attention as in Equation 3.1, yielding as output a d_2-dimensional matrix. This results in h separate output matrices that are concatenated; the final output is then linearly projected to the expected final size. More formally, multi-head attention is defined as

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^0 \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned} \tag{3.2}$$

where the projection matrices have sizes $W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_1}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_2}$ and $W^O \in \mathbb{R}^{hd_2 \times d_{model}}$. In Vaswani et al. [49] we actually have $d_1 = d_2 = d_{model}/h$; then, due to the reduced dimension of each head, the computational cost of a multi-head attention is similar to that of a single attention computed on the full dimensionality d_{model}.

Both encoder and decoder in the Transformer include self-attention layers, where queries, keys, and values all come from the previous layer; however, while every position in the encoder can attend to all positions in the previous layer, in order to preserve the auto-regressive property of the architecture every position in the decoder can only attend to positions before itself. This is realized in practice by masking out (i.e., setting to a very low value) the elements in input to the soft-max which corresponds to illegal connections. Moreover, in the decoder we also have cross-attention layers, where the queries come from the previous decoder layer while the keys and values come from the encoder's output; this allows every position in the decoder to attend to the whole input sequence.

The original Transformer architecture [49] was trained in a supervised manner on a Neural Machine Translation task; later on, BERT (Bidirectional Encoder Representations from Transformers) [9] was introduced as a Transformer-based model making better use of the pre-trained representation for downstream tasks. The authors argue that the unidirectionality of previous LMs leads to sub-optimal results on sentence-level tasks and, more in general, on all those tasks like question answering where incorporating context from both dimensions is crucial. They thus propose a Masked Language Modeling (MLM) pre-training objective, where they randomly mask some tokens in the input and ask the model to reconstruct the original text given the context around the masked words. Since many downstream tasks are based on understanding the relationship between two or more sentences, BERT's authors introduce a second pre-training objective called Next Sentence Prediction (NSP), where they feed to the model the concatenation between two sentences A and B, but B is really the sentence following A in the original text only 50% of the times, while it is randomly chosen in the remaining 50%. Using these two pre-training objectives, along with task-specific fine-tuning, BERT achieves never seen before accuracy on all the tasks from the GLUE benchmark [50] for natural language understanding.

Table 3.2: PLMs for generative tasks in the medical domain.

Model	**Task**	**Loss**
BioGPT [22]	CLM	$\mathcal{L} = -\sum_{t=1}^{T} \log p(x_t \mid \mathbf{x_{<t}})$
BioBART [53]	DAE	$\mathcal{L} = -\sum_{t=1}^{j} \log p(x_t \mid \mathbf{x_{x_{i:j}}}, \mathbf{x_{i:t-1}})$
SciFive [34]	Seq2Seq MLM	$\mathcal{L} = -\sum_{t=1}^{T} \log p(x_t \mid \hat{\mathbf{x}}, \mathbf{x_{<t}})$

Since the main objective of this work is text generation, in the remainder of this Section we will introduce language models that are suitable for our goal; in particular, because we aim to perform data-to-text in the medical domain, we carefully select PLMs that have previously been pre-trained on medical data, like PubMed abstracts or articles. Indeed, a generative model trained on general-purpose documents, when used on medical data, runs the risk of encountering a huge amount of words that it never saw during pre-training. In most cases, generative models use subword-based tokenization algorithms, meaning that they are able to deal with tokens not in the vocabulary by splitting them into sub-words instead of just mapping them to a global OOV (Out-Of-Vocabulary) token as most recurrent models do. Still, having to decompose in sub-tokens a big percentage of the input words in order for the model to process them, might have a pretty big impact on the model's performance. Table 3.2 offers a summary of the models we consider, together with their pre-training objectives; note that $\mathbf{x_{i:j}}$ denotes a masked n-gram spanning from i to j in the sequence $\mathbf{x}$ and $\hat{\mathbf{x}}$ is a randomly perturbed text from the sequence $\mathbf{x}$.

3.1.1 GPT-2

The GPT-2 model [37] is a decoder-only Transformer [49] with masked self-attention heads. The number of layers, the dimension of the hidden space, and the number of heads in the attention mechanism all depend on model size; the various architectures are shown in Table 3.3. With respect to the original GPT [36] work, here the authors modify the position of layer normalization in each block and increase the vocabulary size to 50,257. They also use a context of 1024 tokens, meaning that this model learns to reason on longer spans of text.

Table 3.3: GPT-2 model variations in the HuggingFace's transformers [52] library.

Model Name	**Parameters**	**Layers**	$\mathbf{d_{model}}$	**Heads**
`gpt2`	124M	12	768	12
`gpt2-medium`	355M	24	1024	16
`gpt2-large`	774M	36	1280	20
`gpt2-xl`	1558M	48	1600	25

GPT-2 is trained using a simple Causal Language Modeling (CLM) objective, where the network is asked to predict the next token, knowing everything that came before it in the text. The dataset used for the pre-training step is called WebText and it was realized ad hoc by scraping the information from 45 million Reddit outgoing links.

Thanks to the diverse pre-training dataset, the simple CLM goal actually includes the information and evidence needed to solve a variety of downstream tasks across different domains. Indeed, the authors show that their biggest proposed model, `GPT2-xl`, can obtain state-of-the-art results in a zero-shot setting for a variety of tasks.

In order to achieve this zero-shot capability, the authors focus on realizing a general language model - that is, one which is able to compute the probability of every word and to generate every word. Because common pre-processing steps like lowercasing and the use of OOV tokens restrict the space of strings that can be modeled, they use Byte Pair Encoding (BPE) tokenization [42] applied over byte sequences, which results in a base vocabulary size of just 256. Since the size of the vocabulary is limited, the authors wish to avoid including many variations for a given common word (i.e., *dog.*, *dog?*, *dog!*); they thus prohibit certain types of merges to the algorithm. With this approach, they can effectively model any Unicode string, making their architecture easy to evaluate on any dataset without worrying about the pre-processing steps.

BioGPT [22] is a domain-specific version of `GPT2-medium` for biomedical text generation, which has been pre-trained from scratch using 15 million PubMed abstracts. In particular, the dataset was built by collecting all the PubMed items uploaded before 2021 and filtering out those that had a title but no abstract. The authors also point out the importance of using an in-domain vocabulary, which is not taken from standard GPT-2 but learned from the collected biomedical corpus using BPE; the final size of BioGPT'2 vocabulary is 42,384 tokens. The model is trained with a Causal Language Modeling objective as standard GPT-2 and then tested on biomedical NLP tasks like question answering, end-to-end relation extraction, document classification, and text generation. BioGPT is able to achieve better generation capabilities with respect to standard GPT-2 in the medical domain.

3.1.2 BART

BART (Bidirectional and Auto-Regressive Transformer) [16] follows the standard sequence-to-sequence architecture of the original Transformer model [49]. It is released in two different versions, `bart-base` with 6 layers in both encoder and decoder and a hidden size of 768, and `bart-large` with 12 layers in each component and a hidden size of 1024. BART is a denoising autoencoder and as such, it is trained in two steps: (i) text corruption according to a certain scheme and (ii) optimization of a reconstruction loss between the decoder's output and the original document. Differently from similar architectures, BART is not tied to a specific noising scheme but supports various types of document corruption. Indeed, the authors perform an ablation study on the schemes shown in Figure 3.2 using `bart-base` and find that the best performances are obtained when combining these two noising approaches:

1. *Text infilling*: we mask (using a single `[MASK]` token) arbitrarily long spans of text, including zero-length spans. The model is then asked to reconstruct the original sentence, which forces it to reason on overall sentence structure more than the standard MLM objective does. In practice, the authors mask only about 30% of the tokens in each input element.

2. *Sentence permutation*: the key idea is to randomly shuffle all the sentences of the original text. The authors notice that this noising scheme only produces a significant gain in performance on summarization tasks, but speculate that larger models may have a higher chance of learning significant features from this noising function.

BART can be applied to a variety of downstream tasks: it performs particularly well when fine-tuned for text generation, but this does not come at the expense of its NLU capabilities. Indeed, BART obtains results comparable with RoBERTa [21] on key natural language understanding benchmarks like GLUE [50]. It also outperforms previous state-of-the-art architectures on generative tasks, improving over their results both on CNN/DailyMail [27] summarization - which is suitable for extractive approaches - and on XSum [28] summarization, which is instead highly abstractive. Similar improvements can also be found in tasks like abstractive question answering and dialogue response generation.

BioBART [53] was proposed as a way to address the lack of autoregressive models in biomedical NLP, as the authors point out how most pre-trained architectures in the medical domain are encoder-only transformers with scarce generation capabilities.

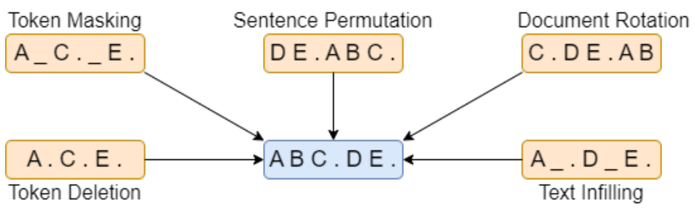


Figure 3.2: Various noising schemes proposed in BART [16].

BioBART is continuously pre-trained on PubMed abstracts, meaning that the model is initialized with the weights from standard BART and then pre-training is continued on domain-specific texts. One major disadvantage of this continual pre-training approach is that the vocabulary is constrained to be the same as the original model; still, it has been shown to be able to achieve comparable performances with respect to the pre-training from scratch approach. Note that the authors perform domain-specific ablation studies on the noising schemes and find sentence permutation to have a negative impact on performance; during continual pre-training, they thus use only text infilling [53].

BioBART is evaluated on various natural language generation tasks in the medical domain (i.e., summarization, dialogue, named entity recognition) and it proves to be a strong in-domain baseline, outperforming the general model.

3.1.3 T5

T5 (Text-To-Text Transfer Transformer) [38] is a standard encoder-decoder Transformer [49] with only minor changes related to the position of layer normalization and to the attention masks; it achieves good performances on both understanding and generation tasks. The baseline T5 model comprises 12 blocks in both encoder and decoder, a hidden space of size 768, and 12 heads in the attention; the authors note how scaling up this base architecture improves performance. However, since there are many low-resource settings where a smaller model is useful, they release a whole family of networks whose details are given in Table 3.4.

The model is pre-trained on a modified version of the data scraped by Common Crawl[1], an open repository of data crawled from the web and devoid of markup and

[1]https://commoncrawl.org/

Table 3.4: T5 model scaling and sizes. d_{ff} represents the dimensionality of the hidden layer in the feed-forward network.

Model Name	**Parameters**	**Layers**	$\mathbf{d_{model}}$	**Heads**	$\mathbf{d_{ff}}$
`t5-small`	60M	6	512	8	2048
`t5-base`	220M	12	768	12	3072
`t5-large`	770M	24	1024	16	4096
`t5-3b`	2.8B	24	1024	32	16384
`t5-11b`	11B	24	1024	128	65536

non-textual content. Unfortunately, while the Common Crawl corpus has been used for various NLP applications throughout the years, most of it is either (i) not really natural language but gibberish and/or standardized templates for menus and error messages or (ii) not useful for the target tasks as it contains offensive language or source code. The authors thus devise come heuristics to clean the data and - because most of their tasks of interest are in English - they also filter based on language. The final dataset is called “Colossal Clean Crawled Corpus” (C4) and, being much bigger than most datasets used for pre-training, allows them to limit/avoid repetitions, which are shown to degrade performance [38].

Self-supervised pre-training is performed using a span-based language masking objective, where we randomly drop 15% of the tokens in the input sequence and replace all consecutive spans of dropped-out tokens with a single sentinel token. These sentinels are special tokens added to the model and do not correspond to any wordpiece. The model is then trained to predict all the dropped-out parts of the text and return them delimited by the same sentinel token we used in the input. An example of this data representation and objective is shown in Figure 3.3; notice how the authors also include a final sentinel token to signal the end of the output sequence.

T5’s paper also points out how, in this text-to-text framework, one can easily realize multi-task learning by mixing various datasets together. Based on this observation, the authors realize multi-task pre-training for their final model simply by considering the unsupervised objective as one of the tasks to be joined together. In this context, the most important factor to decide upon is how much data the model should see for each task: ideally, it should be enough for the LM to learn that task well but not enough to

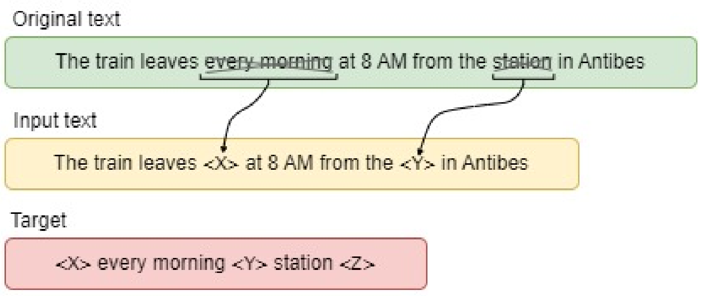

Original text
The train leaves every morning at 8 AM from the station in Antibes
Input text
The train leaves <X> at 8 AM from the <Y> in Antibes
Target
<X> every morning <Y> station <Z>

Figure 3.3: Span-based language masking objective in T5.

memorize the training set. In general, multi-task training alone performs worse than the standard "pre-train and fine-tune" approach. However, if one relaxes the notion of multi-task learning to include a situation where the model is trained on multiple datasets but can still be fine-tuned for the individual tasks later on, the performances become comparable. Plus, with the multi-task approach, we have the advantage of being able to monitor the "downstream" performance throughout the whole training.

T5 can be easily applied to generative tasks, classification/natural language understanding problems, and even regression. It obtains comparable results to those of task-specific architectures and - when scaling up the model - it can also achieve state-of-the-art performances on benchmarks NLU such as GLUE and on abstractive summarization tasks like CNN/Daily Mail.

SciFive [34] is a domain-specific adaptation of T5 [38] for biomedical tasks. In particular, the model is continuously pre-trained starting from the standard checkpoints of `t5-base` and `t5-large` using the original span-based language masking objective. For this phase, the authors combine the C4 dataset with two large biomedical corpora in order to avoid overfitting: specifically, they use abstracts from PubMed and full-text articles from PubMed Central (PMC), a dataset of free articles in the biomedical and life sciences domain. Indeed, they hypothesize that training the model on complete articles may improve its biomedical knowledge while still helping to retain a generalized representation of natural language.

SciFive is then fine-tuned on different medical NLP tasks including question answering, natural language inference, and named entity recognition. The model obtains competitive results on the natural language understanding tasks while reaching state-of-the-art results on generation [34].

3.2 Lightweight Fine-tuning Strategies

As previously stated, fine-tuning is still the main paradigm for adapting large pre-trained language models to downstream tasks, even if it lacks parameter efficiency. It has also been pointed out how, during the fine-tuning phase, generative models may forget important language skills that were acquired through pre-training [12]. Several attempts have been made at alleviating this problem: for instance, He et al. [12] propose a “mix-review” strategy where, for each fine-tuning epoch, the target task data is mixed with a random subset of the pre-training corpus. On a similar wavelength, Chen et al. [5] propose “recall and learn”, where a technique called *pre-training simulation* is used to keep the model’s parameters during fine-tuning close to their pre-trained values; the main idea behind this work is leveraging the multi-task learning paradigm to use the pre-training objective as an auxiliary task during fine-tuning. While both these techniques are able to alleviate forgetting, they do not improve the models’ parameter efficiency.

For this reason, several lightweight fine-tuning approaches have been proposed, often based on the addition of some task-specific modules between the PLM’s layers: these techniques typically freeze the majority of the pre-trained weights, only updating the task-related parameters [19]. In the following Sections, we will explore in more detail the two main approaches to lightweight fine-tuning at the moment, Adapters and Prompt-tuning.

3.2.1 Adapter-tuning

Given a pre-trained Transformer model with parameters θ, Adapter-tuning inserts some task-specific modules with parameters ϕ between its layers; these modules are called *adapters*. Typically, the new parameters ϕ are trained on the target task while keeping the PLM fixed, which implies that only the new weights can learn to encode task-specific features [33].

It has been empirically proven that adding a two-layer bottleneck feed-forward neural network at every layer of the pre-trained Transformer works well [13]; however, deciding exactly where these additional parameters should be placed requires a significant effort as it might have a strong impact on performances [33]. Adapters are usually learned for every downstream task separately and have been shown to obtain results comparable with full fine-tuning while limiting the risk of forgetting. Moreover, they mitigate the problems related to traditional fine-tuning's parameter efficiency in a multi-task scenario, because storing separate copies of millions or billions of weights is not necessary anymore: approximately 99% of the parameters required by a target task are fixed during training, and can be shared across multiple models. This has the added benefit of making it easier to add new objectives to an existing multi-task architecture [33].

It is also interesting to point out how adapters are modular, meaning that we can stack them one on top of the other or dynamically exchange one adapter for another: indeed, since the parameters of the pre-trained language model are fixed, every adapter must learn a representation that is compatible with the following layer on the transformer architecture [33].

3.2.2 Prompting

In its most basic form, the prompting paradigm consists of three steps, (i) prompt addition, (ii) answer search and (iii) answer mapping. First of all, a prompting function $f_{\text{prompt}}(\cdot)$ is applied on the input $\mathbf{x}$ in order to obtain the prompt $\mathbf{x}'$. This function can be thought of as a template, a textual string containing an input slot `[X]` and an answer slot `[Z]`: the first slot is filled with the input text $\mathbf{x}$, while `[Z]` will later contain the generated answer.

For instance, for topic classification with $\mathbf{x}$ = "The Italian team lost the match." a possible template could be "`[X]` This text is about `[Z]`.", meaning that the prompt $\mathbf{x}'$ would become "The Italian team lost the match. This text is about `[Z]`." Other interesting templates for different tasks are shown in Table 3.5; all of them have an empty slot to be filled with the answer $\mathbf{z}$. If `[Z]` is in the middle of the text we talk about a *cloze prompt*, while if the whole text comes before [Z] we talk about *prefix prompts*. In practice, which shape to choose depends on both the task we are solving and the used PLM: prefix prompts work well with auto-regressive LMs and/or for generative tasks, while cloze prompts tend to be preferred when we work with masked LMs. Also notice how, for simplicity, the templates shown in Table 3.5 are all composed of natural language tokens, while in reality they could be composed of "virtual words" (i.e., numeric ids which are later mapped to embeddings) or simply continuous vectors in the parameter space [19].

Table 3.5: Examples of textual prompts and relative answer space for different NLP tasks. Table adapted from [19].

Task Type	**Task**	**Input ([X])**	**Template**	**Answer ([Z])**
Text CLS	Sentiment	My mom loved it.	[X] the product is [Z]	great fantastic ...
	Topic	Our team lost.	[X] the text is about [Z]	food sport ...
Text-pair CLS	NLI	[X1]: An old man... [X2]: A man walks...	[X1]? [Z] [X2]	entailment contradiction neutral
Tagging	NER	[X1]: Al goes to Rome [X2]: Rome	[X1] [X2] is a [Z] entity	location person ...
Text Gen	Summary	OpenAI released...	[X] TL;DR: [Z]	The company... AI models... ...
	Translate	Ça marche	French: [X] English: [Z]	It works. Got it. ...
	Data2text	subj: Belgium rel: language obj: Dutch	[X] Text: [Z]	Dutch is... In Belgium... ..

During the answer search phase, a set $\mathcal{Z}$ of admissible answers is created: for generative tasks, it may be equal to the entire vocabulary of the language model but it is usually a small subset of words. For instance, for the topic classification example, $\mathcal{Z}$ might be defined as {"sports", "science", "politics", "food"}. Then, the correct answer to a prompt is obtained by searching over the set $\mathcal{Z}$ of potential answers as shown in

Equation 3.3, where $h_{\text{fill}}(\mathbf{x}', \mathbf{z})$ indicates the prompt $\mathbf{x}'$ has been filled with answer $\mathbf{z}$ and $P(\cdot)$ indicates the probability of the filled prompt computed using a pre-trained language model.

$$\hat{\mathbf{z}} = \underset{z \in \mathcal{Z}}{\text{search}}\ P(h_{\text{fill}}(\mathbf{x}', \mathbf{z}); \theta) \tag{3.3}$$

Note that the search over the answer space may be a greedy one (i.e., an *argmax* operation) or it might use more complex techniques like beam search or sampling.

The final answer mapping step brings us from the answer $\hat{\mathbf{z}}$ with the highest score to the actual output $\hat{\mathbf{y}}$; this is trivial for generative tasks where answers and output coincide. However, for some classification tasks, the mapping should be explicitly provided, as it may be possible to, for instance, use a variety of different words (i.e., "cooking", "cuisine" or "foodstuff") to refer to a single document topic (i.e., "food") [19].

Figure 3.4 shows some important design decisions for a prompting algorithm; note that since our focus in this work is natural language generation, answer engineering (i.e., the way in which we design Z and, possibly, the mapping function) is not considered. Indeed, we already pointed out how, for generative tasks, Z is usually equal to the entire vocabulary of our model, making this step trivial.

In Section 3.1 we already discussed various pre-trained language models, pointing out their capabilities: every one of those architectures can, in principle, be used to compute the answer's probability in a prompting method. Moreover, we call *prompt engineering* the process of deciding which prompting function f_{prompt} to use in order to obtain the best performance on the downstream task. Figure 3.4 shows how prompt engineering consists of two main choices: the desired prompt shape (i.e., cloze or prefix) and whether to use a manual or an automated approach to create the prompts. Indeed, a very natural way to produce prompts is to manually create them based on our intuition; the LAMA [32] dataset, for instance, provides a set of handcrafted cloze prompts used to probe LM's knowledge. However, manually generating appropriate prompts can be extremely challenging for certain tasks (i.e., semantic parsing), thus automated prompts were proposed. Automated prompts can be discrete or *hard* (i.e., actual text strings) as in AutoPrompt [44], where the original input is combined with a set of learnable trigger tokens according to a template. Since the prompts do not need to be human

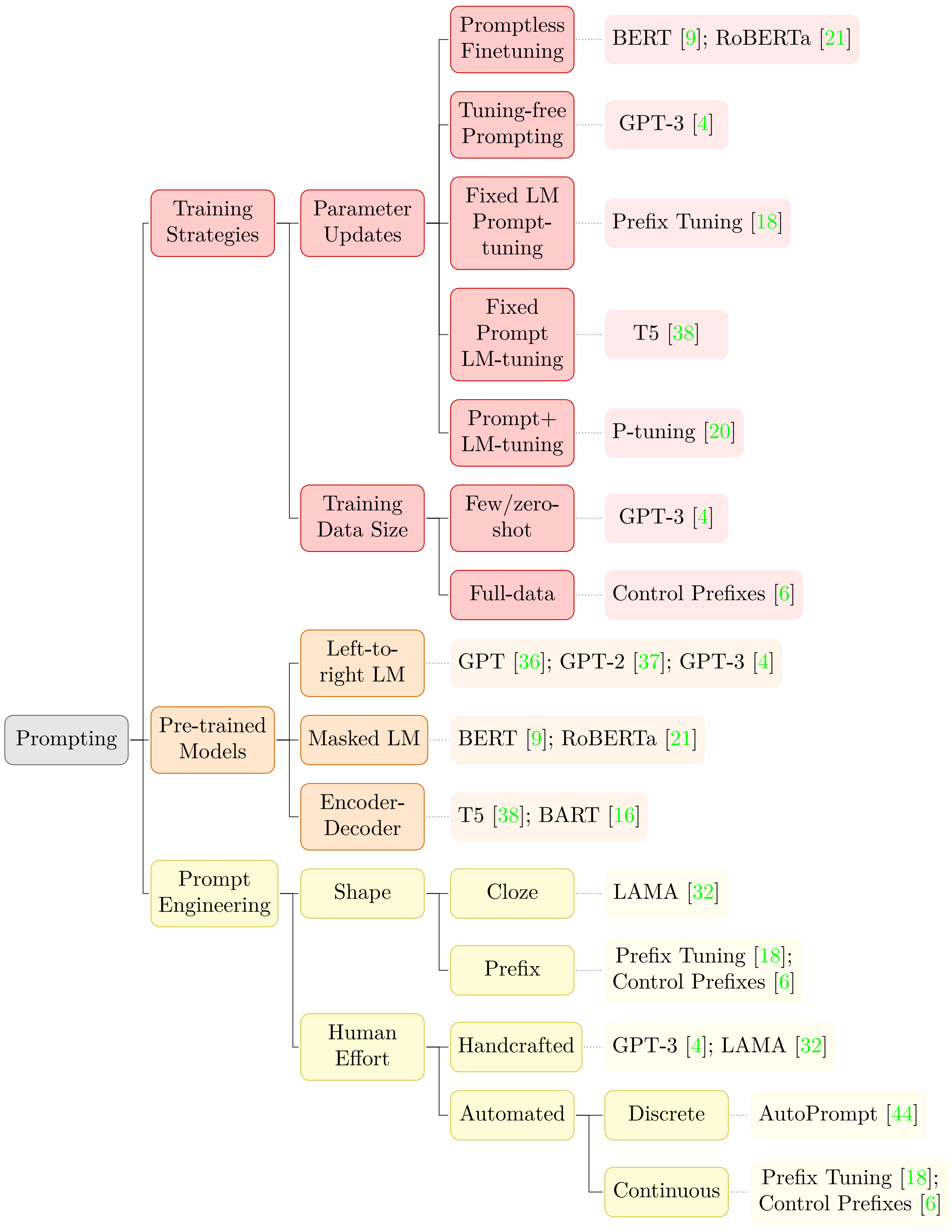


Figure 3.4: Design decisions in prompting algorithms. Image adapted from [19].

interpretable, some methods use continuous (or *soft*) templates: they add new weights to the LM which are directly responsible for parametrizing the prompt, meaning that prompt tokens are not in natural language anymore.

Finally, Figure 3.4 also shows that different prompting algorithms admit different training regimes, both regarding the amount of training data (i.e., the full dataset or zero/few-shot learning) and the number of parameters updated. In particular, we should note that the "fixed-LM prompt tuning" regime, which is mainly used for automated continuous prompts, keeps the LM weights frozen and only trains the prompt parameters. This method can be seen as a sort of lightweight fine-tuning strategy, as it improves parameter efficiency and solves the forgetting problem of traditional fine-tuning. Prefix Tuning [18] and Control Prefixes [6], which will be better described in Section 4, both use this training regime.

3.3 Text Generation

Text generation is an important NLP task that aims to produce coherent and reasonable text in a certain human language. Some common applications of generative techniques are abstractive text summarization, dialogue systems, image captioning, and machine translation. Since generative models are able to learn a way to map inputs to outputs requiring little to none human help, they can also be used to generate free text [17].

Usually, text generation is conditioned on the input data x and to different formats/types of input correspond different instances of the generative task:

1. If x is not provided, the task degenerates into standard language modeling and the output text just needs to be grammatically correct and to sound natural in the chosen language.

2. When x is a sequence of discrete categories (like topic classes), the task becomes controlled generation - meaning that x should steer the content of the output text. This is particularly important in real-world scenarios, as most of them require a way to control certain aspects of the generation. For instance, in the medical domain, we may want to control for the complexity of the answer provided by

the PLM, whose level of technical detail should vary depending on whether it is intended for doctors or patients.

3. When x is a table or any other kind of structured knowledge, the task is called data-to-text and the main objective of the output is to describe x's content in a coherent and accurate manner.

4. When x is an image, a video, or some speech feature we are talking about tasks like image/video captioning or speech recognition. Ideally, the output text should be coherent and faithful to the input.

5. Finally, when x is a text span, we can have multiple possible tasks depending on the application (i.e., machine translation or summarization). Again, the output text is expected to be correct from a grammatical point of view and to respect certain properties which depend on the objective of the downstream task [17].

PLMs for conditional text generation typically output a probability distribution over the whole model's vocabulary for deciding the next word in the sequence, given all the previous ones. Since computing the overall best output sequence is an intractable problem, various decoding strategies have been proposed in recent years [14]. They will be explored in more detail in Section 3.3.1. Finally, notice that automatically evaluating the performance of a generative model is non-trivial as distinguishing correct and incorrect answers often requires knowledge of the target human language, knowledge about the task, and knowledge about the target domain. Popular automatic evaluation metrics for our target task (i.e., data-to-text) are explained in Section 3.3.2

3.3.1 Decoding Methods

As mentioned above, the output of a generative model is a conditional probability of each word in the target text given the input and the previously generated words. Supposing the sequence of tokens generated so far is $y_{<t} = (y_1, \cdots, y_{t-1})$, the model outputs the following probability distribution $P(y_t = w_i | y_{<t}, \mathbf{x})$ for $\forall\ i \in \{1, \ldots, V\}$ with V the size of the vocabulary. A decoding strategy in general aims to obtain the most likely output

sentence overall, i.e., to pick sentence $\mathbf{y}^*$ such that

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \prod_t P(y_t|y_{<t}, \mathbf{x}) \tag{3.4}$$

However, finding such an optimal decoded sequence is an intractable problem, and no sub-exponential algorithms exist to solve it [14]. The easiest way to overcome this problem is to use a greedy algorithm, picking the most probable word at every step; this is a deterministic approach and typically produces short repetitive sentences [14]. Other, more sophisticated, alternatives have been proposed: we particularly focus on beam search as it is currently used in [6] to achieve state-of-the-art performances on the data-to-text task. We also consider contrastive search [46], a recently proposed decoding strategy aiming to make the generated text more natural and less repetitive.

Beam Search At every decoding step, this method keeps track of the most likely b tokens and, only at the end it selects the sequence with the overall highest probability. Usually, the scoring function for a partial sequence is its log-likelihood. Beam search improves over greedy search but can still include repetitions and only explores a limited portion of the search space. This means that, when we use beam search to generate multiple outputs, we often obtain slight variations of the same highly probable sentence, usually involving changes in punctuation or the exchange of a word with a synonym. Nevertheless, for tasks like data-to-text, which do not require high diversity in the generated outputs, beam search and its variants currently achieve great performances.

Contrastive Search This strategy was proposed to alleviate the degeneration problem - that is, the tendency of decoding algorithms to produce dull and repetitive sentences. As briefly mentioned before, this problem is particularly relevant for open-ended text generation tasks like poetry or open-domain dialogue generation; indeed, contrastive search has been shown to significantly outperform other decoding strategies on these kinds of tasks [46], while it has not been evaluated on conditional generation tasks like data-to-text. At every decoding step, contrastive search selects the new token

y_t as

$$y_t = \arg\max_{v \in V^{(k)}} \left\{ (1-\alpha) \cdot \underbrace{P(v|y_{<t}, \mathbf{x})}_{\text{model confidence}} - \alpha \cdot \underbrace{(\max\{s(h_v, h_{y_j}) : 1 \leq j \leq t-1\})}_{\text{degeneration penalty}} \right\} \tag{3.5}$$

where $V^{(k)}$ is the set of top-k prediction of the language model at step t, $\mathbf{x}$ is the input fed to the model and $s(\cdot)$ is the cosine similarity between tokens. The term *degeneration penalty* in Equation 3.5 uses the cosine similarity between token representations to measure how discriminative the new candidate v is with respect to the previous context. Intuitively, a high degeneration penalty means that v is quite similar to the context, leading to repetitive content that - at least in open-ended text generation - should be avoided. Notice that we have two main hyper-parameters: k, which has the same role as b in beam search and is typically chosen in the interval $[3, 10]$, and $\alpha \in [0, 1]$, which regulates the relative importance of degeneration penalty and model confidence [46].

3.3.2 Evaluation Metrics

As previously mentioned, automatically evaluating the quality of a text generation model is non-trivial. Particularly, we are interested in data-to-text tasks, where the input fed to the PLM is structured or semi-structured (i.e., a table or a knowledge graph): here a good evaluation metric should check that the text is fluent, covers all the information provided by the data source and does not hallucinate knowledge. Consider the example in Table 3.6 with three possible generation outcomes: the first one is correct and coherent, but it is missing the birthplace information, while the second one contains a wrong date of birth. The third option is grammatically and factually correct and it contains all the relevant information, but its wording is quite different with respect to the reference sentence [41]. Ideally, a good metric should more strongly penalize the generated sentences containing wrong information with respect to those that are missing some details; sentences that cover all the relevant details in a faithful way but just use different wordings with respect to the reference should suffer from a very small penalization.

The most common metrics used to evaluate data-to-text models are word-based,

Table 3.6: Example of input triplets and generated output for the data-to-text task. Adapted from [41].

Triplet	(John E Blaha, birthdate, 1942 08 26) (John E Blaha, birthplace, San Antonio) (John E Blaha, occupation, Fighter Pilot)
Reference	John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot
Generated	1. John E Blaha who worked as a fighter pilot was born on 26.08.1942. 2. Fighter pilot John E Blaha was born in San Antonio on the 26th July 1942. 3. John E Blaha, born on the 26th of August 1942 in San Antonio, served as a fighter pilot.

meaning that they either (i) interpret both reference and generated sentence as a bag of words or as an n-gram list, then assign a score to the generated text based on the overlap between the two representations or (ii) assign a score to the generated text based on the number of word edits required to make it similar to the reference [41].

BLEU Bilingual Evaluation Understudy (or BLEU) [30] was one of the first automated metrics proposed for text generation, targeting in particular machine translation. It is a precision-based metric that computes the ratio between the number of overlapping n-grams and the total number of n-grams in the generated sentence. If we call $\mathcal{G}$ the set of all the candidate generate sentences, we can then define the modified n-gram precision p_n as

$$p_n = \frac{\sum_{g \in \mathcal{G}} \sum_{\text{n-gram} \in g} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{g \in \mathcal{G}} \sum_{\text{n-gram} \in g} \text{Count}(\text{n-gram})} \tag{3.6}$$

where $\text{Count}_{\text{clip}}(\text{n-gram})$ is the truncated number of occurrences of an n-gram, so as not to exceed its maximum count in a single reference sentence. We can immediately notice that p_n is computed by summing over all the generated sentences: indeed, BLEU is a corpus-level metric, as it returns a single score for the whole dataset [41].

Once p_n has been computed for different values of $n = 1, \ldots, N$, we compute the actual score BLEU-N by taking a weighted mean of those values with uniform weights w_n and multiplying it by a brevity penalty as shown in Equation 3.7. Indeed, we would like our generated sentences to have a length similar to the references: longer candidates are already penalized in p_n so we just need to add a brevity penalty BP computed as shown in Equation 3.8, where $|g|$ and $|r|$ are respectively the generated sentence and the reference length [41, 30].

$$\text{BLEU-N} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log P_n\right) \tag{3.7}$$

$$BP = \begin{cases} 1 & \text{if } |g| > |r| \\ e^{(1-|r|/|g|)} & \text{otherwise} \end{cases} \tag{3.8}$$

METEOR BLEU only allows for matches between n-grams and does not take recall into account. To address these drawbacks, Metric for Evaluation of Translation with Explicit ORdering (or METEOR) [3] is proposed. It is based on the F-score and uses a more relaxed matching scheme: in practice, it first performs exact token matching, then matches together words with the same stem, and, finally, checks for synonyms and paraphrases.

Notice how METEOR examines one word at a time, thus not really considering n-gram matches natively. Instead, it uses a *fragmentation penalty* term to reward longer sequences of contiguous unigram matches. To compute this term, it is first necessary to group into chunks the sequences of matched words. The minimum number of chunks (one) is obtained when all the words in the generated sentence match the reference, while the maximum number of chunks - which is equal to the unigram length of the sentence - is obtained when no words in the generation match with those in the reference [41]. Keeping this in mind, METEOR is computed as

$$\begin{gathered} \text{METEOR} = \text{F-SCORE} \cdot (1 - \text{Penalty}) \\ \text{Penalty} = 0.5 \cdot \left[\frac{\#\text{chunks}}{\#\text{matched unigrams}}\right]^3 \end{gathered} \tag{3.9}$$

where the F-score is obtained by using unigram precision and recall computed according to the relaxed matching strategy described above.

TER As briefly mentioned, many automated metrics are also based on Word Error Rate (WER). A first formulation of WER computes the edit distance as the number of insertions, deletions, and substitutions needed to recover the reference starting from the generated text; Equation 3.10 shows how to compute the Word Error Rate according to this definition.

$$\text{WER} = \frac{\#\text{substitutions} + \#\text{insertions} + \#\text{deletions}}{\text{reference length}} \tag{3.10}$$

Notice that this initial formulation of WER relies heavily on the reference sentence. Since we know that there are many ways to express the same concept in natural language, this metric has also been extended to accept multiple references. Moreover, the word order is very important in WER and, every time a unigram is found in a different position in the generation with respect to the reference, we are required to first delete it and then insert it back in the correct place [41]. However, giving such a large weight to a misplaced word might be counter-intuitive, because a natural language sentence can still be correct if the word order changes in some parts. Thus, the alternative Translation Edit Rate (TER) was proposed which adds “word shifting” to the admissible operations [45].

Chapter 4

Generating Text via Prompting

As previously mentioned, text generation tasks aim to produce coherent text in a certain human language, often conditioned on some other information. Prompting techniques can easily be used for generative tasks by combining prefix prompts with auto-regressive LMs [19]. For instance, GPT-3 [4] shows great few-shot capabilities on a wide range of tasks including machine translation and question answering, simply using handcrafted natural language prompts.

Li and Liang [18] point out how having a proper context can steer the LM toward producing the desired output without needing to change its parameters. Supposing we want to generate a certain word x, we know that if we prepend as a context some tokens that often appear together with it, then the LM will assign a much higher probability to x. This can be intuitively extended to the generation of a whole span of text, but how to choose the context is non-trivial: preliminary experiments show that, while natural language instructions may help a human annotator and can be used to steer LMs like GPT-3 that have hundreds of billions of parameters, they fail for most smaller language models (i.e., GPT-2 and BART). Li and Liang [18] thus propose to optimize the context as continuous parameters that will be prepended to the input at every layer of the LM. Differently from embedding-only tuning, where we can just optimize the embeddings of the prompt's "virtual tokens" and the upper-level activations are left to be computed by the frozen transformer, in Prefix Tuning we have a tunable prompt element at every layer of the model. The authors show that Prefix Tuning improves performances by a wide margin with respect to embedding-only tuning.

Because Li and Liang [18] show that Prefix Tuning can close the gap with traditional fine-tuning strategies for data-to-text tasks while being more parameter efficient and limiting forgetting, all the approaches we explore in this work are based on it. Section 4.1 goes into more detail on the Prefix Tuning technique, while Sections 4.2 and 4.3 explain two alternative approaches still based on the idea of optimizing a prefix at every level of the transformer - that is, Control Prefixes and Prefix Pooling.

4.1 Prefix Tuning

Prefix Tuning [18] can intuitively be understood as the concatenation of new learnable weights to the attention keys and values of every layer in our architecture, as shown in Figure 4.1. More formally, suppose we have a generative Transformer-based language

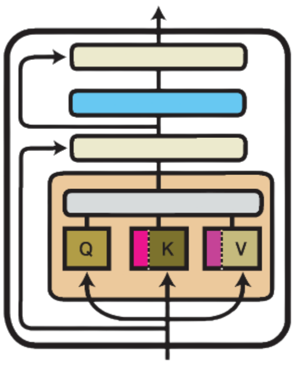


Figure 4.1: Schematized view of a Transformer's layer: learnable weights for Prefix Tuning are shown in pink and purple. Picture from the AdapterHub [33] website.

model $p_\phi(y|x)$ parametrized by ϕ and we denote with d the dimension of the hidden state and with L the number of layers. Then, when computing the attention at layer i, we call $Q_i \in \mathbb{R}^{N\times d}$ the query matrix, and $K_i, V_i \in \mathbb{R}^{M\times d}$ the key and value matrices respectively. Note that N simply indicates the number of tokens of the query, while M is the number of tokens in key and value.

The idea behind Prefix Tuning is to learn a set of key-value pairs $P = \{P_1, \cdots, P_L\}$, where $P_i \in \mathbb{R}^{\rho\times 2dL}, \ \ \forall i \in 1\dots L$ and ρ is the length of our prompt - that is, the number of additional key-value pairs we add to every attention computation. In practice, at layer i we augment K_i and V_i as shown in Equation 4.1, obtaining $\hat{K}_i, \hat{V}_i \in \mathbb{R}^{(\rho+M)\times d}$

$$\hat{K}_i = [P_{i,K}; K_i] \qquad \hat{V}_i = [P_{i,V}, V_i] \tag{4.1}$$

Recall that decoder-only models like GPT-2 have a single type of attention (i.e., the masked self-attention in the decoder), thus we learn a single overall general task prefix parametrized by θ that we call P_θ. For encoder-decoder architectures (i.e., BART, T5), the authors find it beneficial to learn a different set of key-value pairs for every attention type: we thus call P^E the prefix for the encoder self-attention, P^C the prefix for the decoder cross-attention and P^M the prefix for the decoder masked attention. The overall general task prefix is still indicated as $P_\theta = \{P^E, P^C, P^M\}$ and parametrized by the free weights θ. The training objective in Prefix Tuning is the same as used in standard fine-tuning, but we freeze the LM parameters ϕ and only tune the prefix θ. This is a sensible lightweight fine-tuning choice because the prefix parameters always influence the hidden states of our model as they are concatenated to the left of all other activations (i.e., they act as a context) [18].

Li and Liang [18] also show that updating directly the prefix parameter's matrix P_θ is particularly sensitive to the choice of learning rate and initialization. The optimization step can be stabilized by over-parametrizing the prefix through the use of a two-layer feed-forward network. This increases the number of learnable parameters, thus making our model less parameter-efficient at training time: however, the reparametrization can be dropped after training is concluded and we only need to store P_θ for inference.

Prefix Tuning outperforms adapter-tuning and other lightweight fine-tuning strategies on the data-to-text generation task, showing performances comparable with standard fine-tuning and even outperforming it in low-data settings [18]. Moreover, this technique can generalize well to different domains and is easy to scale up with the size of the fixed language model. The length ρ of the prefix is a tunable hyper-parameter and the authors note that increasing it results in improved performances up to a certain threshold: this is reasonable because a longer prefix implies a greater amount of parameters, which make our architecture more expressive [18].

4.1.1 A different parametrization

Despite being simple and effective, Li and Liang's parametrization of the prompt via a two-layer feed-forward network has a small drawback. Indeed, the output of the MLP (with shape $\rho \times 2dL$) already represents prefix keys and values for every layer of the model, implying that the prefix at layer $i + 1$ is computed without an explicit dependency on the prefix at layer i. Since features computed by a network at each layer depend on lower-level features coming from previous layers, we argue that introducing this prior in the prefix parametrization could be beneficial. Accordingly, for every layer $i+1$ we use as a prefix a weighted sum between its value and that of the previous layer as estimated by the MLP, i.e., $\hat{P}_{i+1} = P_{i+1} + w_i P_i$. Note that this approach only adds $L - 1$ trainable parameters to the model and, while seemingly naive, may effectively improve performance as we show in Section 6.

4.2 Control Prefixes

Prefix Tuning achieves great generation capabilities and is able to close the gap with traditional fine-tuning for the data-to-text task; however, it just prepends a general task prefix to all the input examples, thus leaving no way to incorporate attribute-level information and/or control the generation. The controllability factor is particularly relevant in the medical domain, as we might want to be able to steer the content of the generated explanations and their level of technical details.

To address these issues, Clive et al. [6] proposed Control Prefixes, a dynamic prompting technique that extends Prefix Tuning where example-specific information can act as a guidance signal. In particular, this additional information influences the choice of a second prefix, which gets concatenated with the static task-specific prompt from Prefix Tuning. This second modular prefix can operate together with the static one to guarantee more fine-grained control over the generation, while the underlying pre-trained language model stays frozen. A comparison between this technique and Prefix Tuning can be found in Figure 4.2; note that the guidance signal can either provide new information about the input (i.e., the domain of a set of triplets) or specify a desired

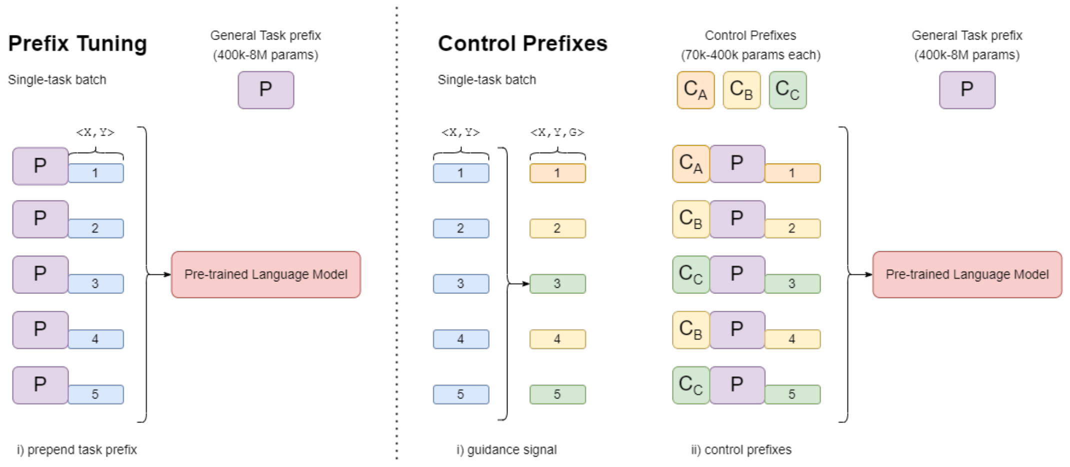


Figure 4.2: Comparison between Prefix Tuning and Control Prefixes for a single-task batch. Image adapted from [6].

property for the output (i.e., the length or the level of details of the explanation).

More formally, this technique assumes that the training corpus contains at least one guidance attribute G_1 with R possible discrete labels. In addition to the task-specific prefix P_θ defined as in Section 4.1, the model now also includes also a set of control prefixes $C_\theta = \{C_{\theta,1}, \dots, C_{\theta,R}\}$, where $C_{\theta,k} \in \mathbb{R}^{\rho_c \times 2dL}$ represents the set of parameters learned for value k of our guidance attribute G_1. Note that d still indicates the dimension of the model's hidden state, L still represents the number of layers, and ρ_c denotes the length of the dynamic control prefix for the particular attribute G_1. Accordingly, Equation 4.2 shows the new keys and values for the attention at layer i of our model; note that $\mathcal{F}(\cdot)$ is the function which takes the attribute label indicated by G_1 and returns the respective set of prefix vectors.

$$\hat{K}_i = [\mathcal{F}(G_1)_{i,K}; P_{i,K}; K_i] \qquad \hat{V}_i = [\mathcal{F}(G_1)_{i,V}; P_{i,V}; V_i] \tag{4.2}$$

Notice how the modular control prefixes act as a left context for both the model's original key and values and the parameters of the static, task-specific, prefix. This means that the task-specific parameters can somehow adapt themselves to the control prefixes, making it possible for the fine-grained control over the generation to also help downstream task performance [6].

Similarly to Prefix Tuning [18], in encoder-decoder models we again have different control prefixes for every type of attention, meaning that every prefix $C_{\theta,k}$ comprises of three constituents $C_{\theta,k} = \{C_k^E, C_k^C, C_k^M\}$. To stabilize the optimization process we again need to employ a re-parametrization: the authors find it beneficial to share it with the static task-specific prefix P_θ and point out that, otherwise, the number of learnable weights at training time would increase dramatically, leading to reduced performance.

When using the triplet categories as guidance signal in a data-to-text task, Clive et al. [6] show that their method can more effectively capture some of the input's properties while improving over both previous state-of-the-art fine-tuning strategies and Prefix Tuning. Control Prefixes can also be used to perform zero-shot learning: indeed, we can expect that, when attribute labels are semantically similar, their respective control prefixes have a similar impact on the task-specific prefix and frozen language model. Under this assumption, when dealing with unseen test-time categories (i.e., those that were not part of the training set), we can obtain a good zero-shot performance by mapping them to their GloVe [31] embeddings and computing the cosine similarity in the embedding space between every train label and every unseen category. We finally select as control prefix for an example with previously unseen category, the one corresponding to the most similar training label [6].

4.3 Prefix Pooling

We said that, while task-specific prefixes (i.e., Prefix Tuning [18]) can close the gap with traditional fine-tuning techniques, they are static and do not depend in any way on the input example. We however argue that, in the same task, we might have examples of different complexity: for instance, in data-to-text, it's much harder to produce a coherent description that is neither missing nor hallucinating information for long triplet sets, while examples with just one or two triplets in the input set are almost trivial. Based on this observation, we might want our prefix, or at least a part of it, to change depending on the input. Control Prefixes [6] are a dynamic prompting technique that may seem a good solution to our problem. Recall however that, while they do work well in a zero-shot setting at inference time, the additional guidance attribute must be present during training for this approach to be employed.

In this work we thus propose Prefix Pooling, a dynamic prompting technique that does not make use of any guidance signal extraneous from the input text; we show the intuition behind this approach in Figure 4.3. Inspired by Wang et al. [51], we design

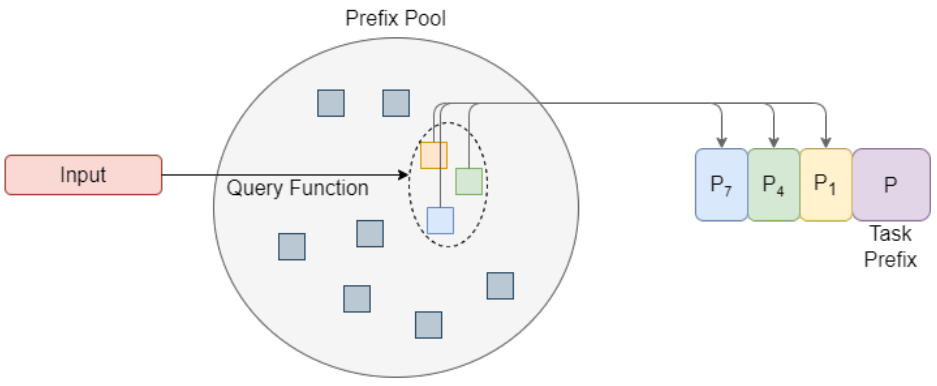


Figure 4.3: Intuition behind the Prefix Pooling dynamic prompting approach.

a pool of E prefixes $\{P_{\theta,1}, \cdots, P_{\theta,E}\}$, each one of length m and associated to a key k_i, which can either have a fixed value (i.e., the mean/max of the m vectors in $P_{\theta,i}$) or be a learned vector. Then, every input example is fed to the embedding layer of our frozen PLM, and a query vector is computed starting from the obtained embedding matrix; similarly to what we said for the keys, the query vector could simply be the mean/max over the matrix or it could be computed through more sophisticated approaches like a small MLP. Once we have a set of keys and a query, we compute the similarity between them, pick the n keys with the highest value and concatenate the corresponding prefixes to the task-specific prompt. More formally, if $\mathcal{G}(\cdot)$ is the function which, starting from the input ids X returns the top n prefixes according to the process described above, the keys and values of the attention for layer i of our model become the following, where $P_{i_K}, P_{i,V}$ are the standard task-specific prefixes defined in Section 4.1.

$$\hat{K}_i = [\mathcal{G}(X)_{i,K}; P_{i,K}; K_i] \qquad \hat{V}_i = [\mathcal{G}(X)_{i,V}; P_{i,V}; V_i] \tag{4.3}$$

Note that the number of prefixes E in the pool, the length m of those prefixes, and the number n of dynamic prefixes to concatenate to the task-specific prompt are all tunable hyper-parameters of our model. Moreover, since both prefixes and keys are randomly initialized, to ensure that all the weights get updated, in 30% of the cases during training we swap the actual top-n selected prefixes for some random ones.

Following both Prefix Tuning [18] and Control Prefixes [6] the prompts are not directly optimized but we employ a two-layer MLP re-parametrization, which can be dropped once training is concluded. Also note that in encoder-decoder models we use three different pools, one for every type of attention.

Chapter 5

Datasets

In this work, we strive to solve a data-to-text task in the medical domain, producing high-quality descriptions of clinically-oriented triplet sets. As motivated in Chapters 3 and 4, we choose to employ state-of-the-art prompting techniques and we propose a novel dynamic prompting approach. Because of this, we also use a general domain benchmark to provide fair comparison with previous research: particularly, we work with the popular WebNLG [10] benchmark for neural data-to-text generation, which we introduce in Section 5.1. Focusing on the medical domain, our triplets need to contain information about the symptoms reported in a clinical case and about the related correct disease. Then, in line with the scope of the ANTIDOTE project, the generated description can be used to interpret automatic diagnoses in a post-hoc[1] fashion. To the best of our knowledge, no dataset is available to perform data-to-text on clinical data that satisfies these requirements. In Section 5.2 we thus present USMLE-Symp, a corpus that was created ad hoc using the previously annotated ANTIDOTE [1] data and the medical information coming from the HPO [15] ontology.

Notice that both datasets are analyzed under the same settings: we report information about the input size in characters, words, and number of triplets for the whole dataset. We report information about the reference descriptions' length in words and characters only for the train split. Throughout the whole analysis, we consider as a character any symbol in the text, once leading and trailing spaces have been removed. We instead consider as a word any entry of the text after dividing camel case words

[1]Post-hoc explanations start from a previously trained black-box AI model and try to produce a human-understandable representation of its inner workings.

(i.e., “birthPlace” becomes “birth Place”) and performing tokenization based on whitespaces, hyphens/dashes, and underscores.

5.1 WebNLG

The WebNLG corpus maps RDF (Resource Description Format) triples extracted from DBpedia [2] to a textual description in English. DBpedia is a multilingual knowledge base built starting from Wikipedia’s structured information. This data is stored as triplets of the format `(subject, property, object)` where the `subject` is a Uniform Resource Identifier (URI), a sequence of characters that identifies a resource uniquely and universally. The `property` is simply a binary relation and the `object` is either another URI or a literal value (i.e., a string, a date, or a number). The extracted sets of triplets are annotated via crowdsourcing following four main steps:

1. Clarifying properties: the meaning of a DBpedia property may be unclear without consulting the documentation, thus they are manually changed to easier-to-verbalize synonyms.

2. Getting verbalizations for triplets sets of size one: three separate descriptions are collected for data units with size one (i.e., single triplets). In this phase, both automatic and manual checks are employed to ensure data quality.

3. Getting verbalizations for longer triplet sets: the annotators are asked to merge together descriptions corresponding to single triplets to form natural-sounding verbalizations. Building upon the single-triplet descriptions helps reducing the risk of misinterpreting the original meaning of that data unit.

4. Quality verification: the quality of the descriptions obtained at step 3 is verified using crowdsourcing. The participants are asked to assess their fluency, semantic adequacy with respect to the original triplet, and grammatical correctness.

Table 5.1 shows some qualitative examples of the triplet sets and relative verbalizations obtained following the procedure above. In particular, for this work we used the

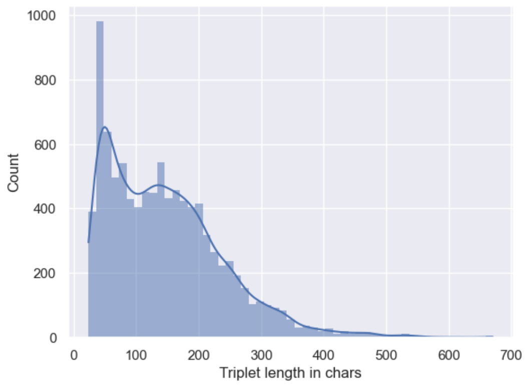
Table 5.1: WebNLG dataset: qualitative examples.

Triplets	(Aarhus_Airport, cityServed, “Aarhus, Denmark”)
References	1. The Aarhus is the airport of Aarhus, Denmark. 2. Aarhus Airport serves the city of Aarhus, Denmark.
Triplets	(Buzz_Aldrin, dateOfRetirement, “1971-07-01”), (Buzz_Aldrin, timeInSpace, “52.0”(minutes))
References	1. Buzz Aldrin, who retired on July 1, 1971, spent 52.0 minutes in space. 2. Buzz Aldrin spent 52 minutes in outer space before he retired on January 7th, 1971. 3. Buzz Aldrin, who retired on July 1st 1971, once spent 52 minutes in outer space.
Triplets	(A.S._Roma, ground, Stadio_Olimpico), (A.S._Roma, league, Serie_A)
References	1. AS Roma, in the Serie A league, has its grounds in Stadio Olimpico. 2. A.S. Roma play in Serie A and their ground is the Stadio Olimpico. 3. A.S Roma’s ground is Stadio Olimpico, and they play in the Serie A league
Triplets	(A_Wizard_of_Mars, language, English_language), (English_language, spokenIn, Great_Britain), (United_States, capital, Washington,_D.C.), (A_Wizard_of_Mars, country, United_States)
References	1. Washington, D.C. is the capital of the United States where A Wizard of Mars originates. This novel is published in English which is the language spoken in Great Britain. 2. A Wizard of Mars was Published in the United States. The book is written in English (originated in Great Britain). The capital of the US is Washington D.C.
Triplets	(14th_New_Jersey_Volunteer_Infantry_Monument, category, Historic_districts_in_the_United_States), (14th_New_Jersey_Volunteer_Infantry_Monument, district, Monocacy_National_Battlefield),(14th_New_Jersey_Volunteer_Infantry_Monument, established, “1907-07-11”)
References	1. The 14th New Jersey Infantry Monument which is located in the Monocacy National Battlefield was established on 11 July 1907. It falls within the category of Historic districts in the US. 2. Monocacy National Battlefield is the location of the 14th New Jersey Volunteer Infantry monument which was established on 11 July 1907 and belongs to the category of historic districts in the US. 3. The 14th New Jersey Volunteer Infantry Monument was established on 11 July 1907 and is located in the Monocacy National Battlefield. being a historic district in the United States.

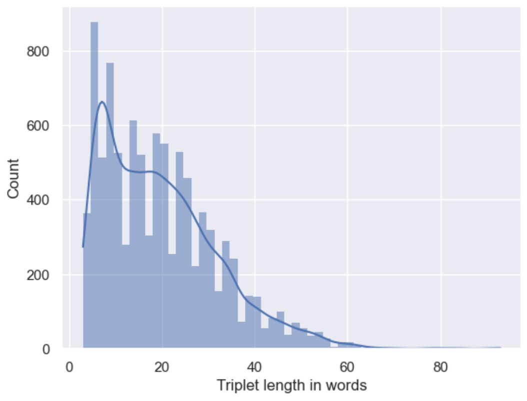
HuggingFace hosted[2] version of the WebNLG 2017 Challenge dataset, which consists of 25,212 data/text pairs with a total of 9,669 distinct input triplet sets. The training and validation data describe entities coming from 10 distinct DBpedia categories (i.e., Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, City, and WrittenWork). The test dataset also includes examples belonging to

[2]https://huggingface.co/datasets/web_nlg/viewer/webnlg_challenge_2017

five additional *unseen* categories, which can be used to estimate our models' generalization capabilities to different domains. Figure 5.1 shows the distribution of the triple set's length in characters and words for the whole dataset. We can see that most inputs are shorter than 600 characters and 60 words, respectively. Since every model we are planning to use has its own (often sub-word based) tokenizer, these values only give us an idea of the effective number of tokens the input will have in practice. We still argue that the low estimations indicate that there is no need to truncate the triplet sets, and potentially lose meaningful information, to abide by the maximum length each model can process (i.e., usually 512 tokens). Similarly, Figure 5.2 shows the input size in



(a) Distribution of input length in characters.



(b) Distribution of input length in words.

Figure 5.1: Histograms of input triplet sets' length in characters and words for the whole WebNLG dataset.

number of triplets for the whole dataset: we can see how most inputs are composed of up to 5 triples while only few of them have 6 or 7 components. This is important because, as previously mentioned, the more triplets we need to verbalize the harder it is to get a coherent description that contains all necessary information without hallucinating. Next, we investigate whether there is any correlation between the extent of the input and its category: because the length distributions in characters and words are very similar, for this analysis we only consider input size in words and number of components. Figure 5.3 shows our results: we can clearly see that some categories (i.e., University, Astronaut) tend to have inputs with more triplets, which can also be assumed to be more complex to verbalize. Moreover, it is quite clear that, in general, a bigger number of words implies a bigger number of triplets in the data unit. There might however be exceptions to this rule: for instance, the triplets might be composed of long and repeated `subjects` or `objects`. This is clearly the case for the last example of Table 5.1 that contains more words and characters than the second-to-last example while being composed of one less triplet.

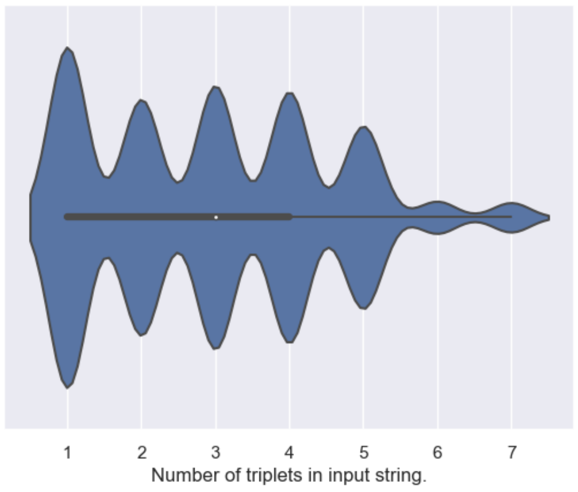
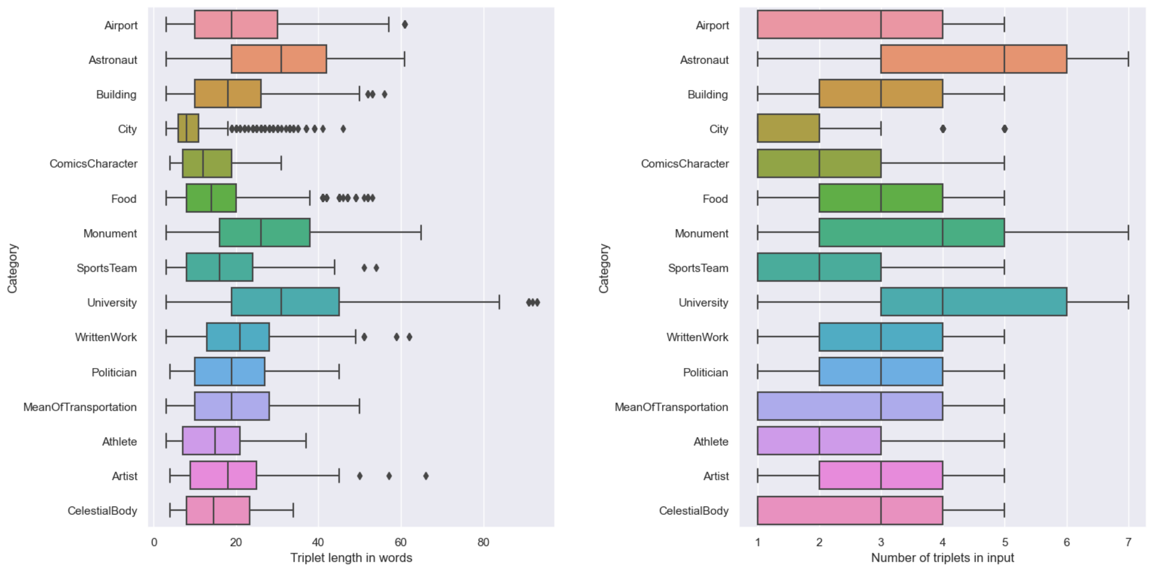


Figure 5.2: WebNLG input size in number of components for the whole dataset.



(a) Category-wise input length in words. (b) Category-wise input size in triples' number.

Figure 5.3: Box plot of category-wise input size in words and number of components. The plot refers to the whole WebNLG dataset.

Finally, Table 5.2 shows some summary statistics related to the length (in both words and characters) of the input and of the target verbalizations. Note that the information about the descriptions is only computed on the training set. We can see that neither the input nor the train verbalizations ever go below 3 words, while their mean length are 19.75 ± 12.46 and 19.84 ± 10.20 respectively. Finally, notice that we have a lot of variability in the size of both input and target sentence.

Table 5.2: WebNLG summary statistics for both input and verbalizations.

	Type	**Mean**	**Min**	**Max**	**Std**
Triplet	chars	147.96	24.00	671.00	95.41
	words	19.75	3.00	93.00	12.46
Verbalization	chars	117.08	20.00	445.00	60.71
	words	19.84	3.00	72.00	10.20

5.2 USMLE-Symp

The USMLE-Symp dataset is built specifically for our clinical data-to-text task: as explained in Chapter 2, a system generating template-based explanations for the diagnosis associated to a USMLE[3] clinical case was already produced in the context of the ANTIDOTE project. This work aims to improve those natural language explanations by moving away from single templates: in particular, we frame the problem as a data-to-text task and strive to solve it using pre-traiend language models. In order to do this, we first need to generate the dataset starting from the following four types of previously extracted information:

- Case symptoms: all the symptoms that are detected from the USMLE case description. We use this information to build triplets of the type `(patient, Has_Symptom, symptom)` or `(patient, is, symptom)` if the symptom is actually an adjective like *confused.*
- Patient diagnosis: we use the information about correct and incorrect diagnoses for our clinical case in order to build triplets of the following types, `(patient,`

[3]The MedQA-USMLE is a multiple choice question answering dataset collected from professional medical board exams. Previous work in the context of the ANTIDOTE project focused on annotating part of it with symptoms, population groups, and findings in order to train an automated extraction system.

`Suffers_from, correctDisease)` and `(patient, Does_not_suffer_from, incorrectDisease)`.

- Symptoms associated with a disease: for every disease for which we find a match in the HPO ontology, we extract the related symptoms and build triplets of the type `(disease, Has_as_symptom, symptomName)`.
- Patient information: the details about age group and gender of the patient, which is used to build triplets of the type `(patient, is_A, populationGroup)`. Note that to introduce variety in our corpus we sometimes decide to omit the aforementioned triplet and just substitute `patient` with `populationGroup` in all the other components. This means that if we know that the patient is a 32 years old woman and that she is confused we might generate either the set of triplets `(patient, is_A, 32 years old woman)`, `(patient, is, confused)` or the single triplet `(32 years old woman, is, confused)`.

Considering all these triplets together for a certain clinical case, we can obtain the relative *why* or *why-not* explanations, whose verbalization is given by the templates shown in Table 2.1. Because it is likely that the same concept can be described in multiple ways, we produce additional reference target texts using the online paraphrasing tool QuillBot[4]. We also tried to use language models fine-tuned for text rewording but we obtained lower-quality results, often missing crucial information. Table 5.3 shows the final templates we use to verbalize both explanation types for our dataset.

We previously pointed out how, in benchmark data-to-text datasets like WebNLG, data units are composed of different amounts of triplets, ranging from simple 1-component examples to more complex inputs with 6 or 7 elements. In order to mimic this structure for our clinical data, we also generate multiple template verbalizations for every single triplet containing only patient and/or disease information. We then group them together to obtain data units of intermediate sizes (i.e., 4-7 triplets). Table 5.4 shows some qualitative examples of the triplet sets and relative verbalizations obtained through this procedure; notice how the textual descriptions often appear less fluent than the

[4]https://quillbot.com/

Table 5.3: Template reference verbalization for why and why-not explanations.

Why	1. The patient is showing [DISEASE] as the following symptoms are direct symptoms of [DISEASE] and appear in the case description: [SYMPTOMS].
	2. The patient can be diagnosed with [DISEASE] as they are showing the following symptoms, which are direct symptoms of [DISEASE]: [SYMPTOMS].
	3. The following symptoms, [SYMPTOMS], which are direct symptoms of [DISEASE], indicate that the patient has that disease.
	4. These symptoms, [SYMPTOMS], are indicative of the disease [DISEASE], which is thus the correct diagnosis.
	5. The patient is exhibiting symptoms like [SYMPTOMS], which are direct signs of [DISEASE]. The patient should thus be diagnosed with [DISEASE].
Why not	1. As for the [DISEASE] diagnosis, it has to be discarded because the patient is not showing [SYMPTOMS], symptoms that cannot be found in the case description.
	2. The patient is not exhibiting [SYMPTOMS], hence the [DISEASE] diagnosis must be rejected.
	3. The following list of symptoms is not found in the case description, hence the [DISEASE] diagnosis must be rejected: [SYMPTOMS].
	4. Due to the patient's lack of certain symptoms, which are not mentioned in the case description, the [DISEASE] diagnosis has to be rejected. The missing symptoms are: [SYMPTOMS].
	5. [DISEASE] has been rejected as a possible diagnosis for the patient due to the lack of the following symptoms: [SYMPTOMS].
	6. The patient is not exhibiting these symptoms: [SYMPTOMS]. Since they are signs of [DISEASE], that diagnosis must be discarded.

WebNLG ones, as well as less grammatically correct. This is to be expected since the WebNLG data was crowdsourced while ours is mostly based on paraphrased templates. The final dataset has 10,938 data/text pairs and a total of 5,353 distinct input units; it is split into a training set containing 3,735 triplets and 7,596 input/target pairs, a test set composed of 916 data units, and a validation set of size 702. In order to associate category information to every example in our dataset, we extract the most relevant symptom or disease from it and search for its classification in the HPO ontology. When no match is found in HPO, category information for a disease/symptom is selected to be the same as the one of its closest neighbor in the embedding space, where the embeddings are computed through a pre-trained PubMedBERT[5] model. We also add polarity information to our dataset, that is, for every example, we label it as *positive* if it states that a patient has a certain disease/symptom and as *negative* otherwise. We aim to use this information to steer the language model toward producing effective explanations

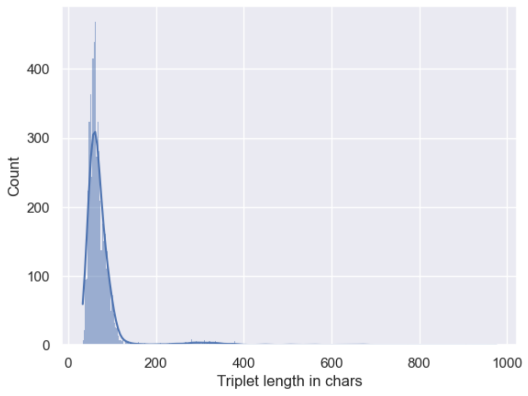
[5]https://huggingface.co/microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract-fulltext

for the desired situation. That is, we do not want the language model to produce a "why" description when we are trying to justify a negative diagnosis or vice-versa (i.e., a "why-not" explanation to a positive diagnosis).

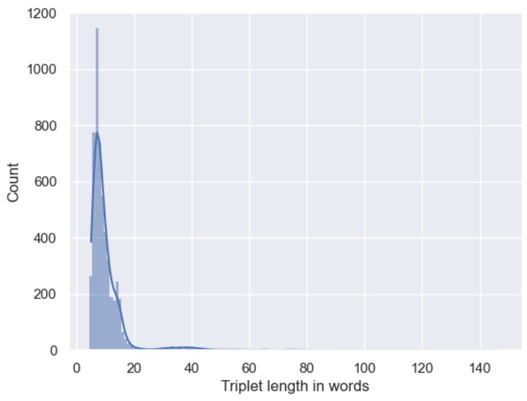
Table 5.4: USMLE-Symp dataset: qualitative examples.

Triplets	(patient, is_a , 34-year-old woman), (patient, Suffers_from, Thrombotic thrombocytopenic purpura)
References	1. The patient, a 34-year-old woman, has been diagnosed with Thrombotic thrombocytopenic purpura. 2. The patient is a 34-year-old woman and suffers from Thrombotic thrombocytopenic purpura.
Triplets	(Beta-thalassemia major, Has_as_symptom, Upslanted palpebral fissure)
References	1. Upslanted palpebral fissure is a symptom of Beta-thalassemia major. 2. People with the disease Beta-thalassemia major may show Upslanted palpebral fissure as a symptom.]
Triplets	(Bullous pemphigoid, Has_as_symptom, Diabetes mellitus), (Bullous pemphigoid, Has_as_symptom, Recurrent infections), (Bullous pemphigoid, Has_as_symptom, Autoimmunity), (Bullous pemphigoid, Has_as_symptom, Eczema), (Bullous pemphigoid, Has_as_symptom, Urticaria)
References	1. The disease Bullous pemphigoid is typically characterized by these symptoms: Diabetes mellitus, Recurrent infections, Autoimmunity, Eczema and Urticaria. 2. These symptoms indicate that the patient my suffer from Bullous pemphigoid: Diabetes mellitus, Recurrent infections, Autoimmunity, Eczema and Urticaria.
Triplets	(patient, is_a, 34-year-old woman), (patient, Has_symptom, fever), (patient, Has_symptom, headache), (patient, is, confused), (patient, is, oriented only to person), (patient, Has_symptom, petechiae)
References	1. The patient, a 34-year-old woman, has the following symptoms: fever, headache and petechiae. The patient is also confused and oriented only to person. 2. The patient is a 34-year-old woman and is affected by these symptoms: fever, headache and petechiae. The patient is also confused and oriented only to person.
Triplets	(von Willebrand disease, Has_as_symptom, Abnormality of thrombocytes) (von Willebrand disease, Has_as_symptom, Abnormality of coagulation), ... (40-year-old woman, Does_not_suffer_from, von Willebrand disease), (40-year-old woman, Has_symptom, fever and confusion), (40-year-old woman, Has_symptom, petechiae)
References	1. The patient is not exhibiting Abnormality of thrombocytes, Abnormality of coagulation, Abnormal platelet function, Abnormal mitral valve morphology and Venous insufficiency, hence the von Willebrand disease diagnosis must be rejected. 2. Due to the patient's lack of certain symptoms, which are not mentioned in the case description, the von Willebrand disease diagnosis has to be rejected. The missing symptoms are: Abnormality of thrombocytes, ... 3. von Willebrand disease has been rejected as a possible diagnosis for the patient due to the lack of the following symptoms: Abnormality of thrombocytes, Abnormality of coagulation, ...

Figure 5.4 shows the distribution of the triplet set length in both characters and words for the whole dataset. We can see that most inputs are shorter than 200 characters and 20 words respectively, although there are some outliers that appear far longer. As pointed out for WebNLG in Section 5.1, these values are only an estimation of the effective length our input will have once processed by the appropriate tokenizer, but they are enough to state that we will not need to truncate our input before feeding it to the model. Figure 5.5 instead shows the input size in number of triplets for the



(a) Distribution of input length in characters.



(b) Distribution of input length in words.

Figure 5.4: Histograms of input triplet sets' length in characters and words for the whole USMLE-Symp dataset.

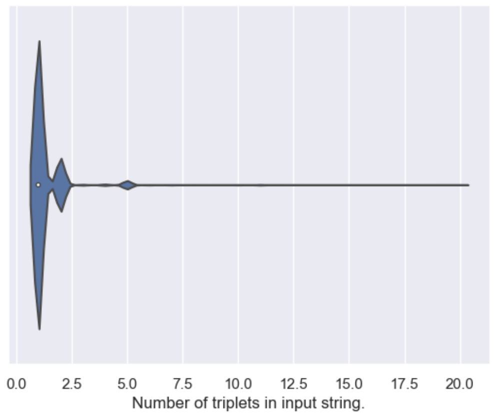
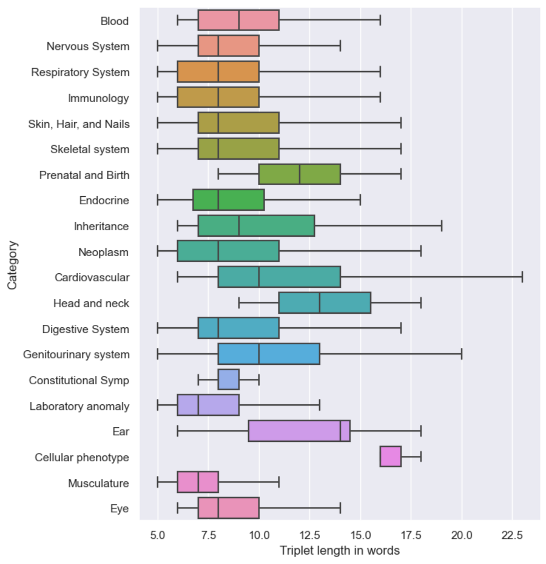
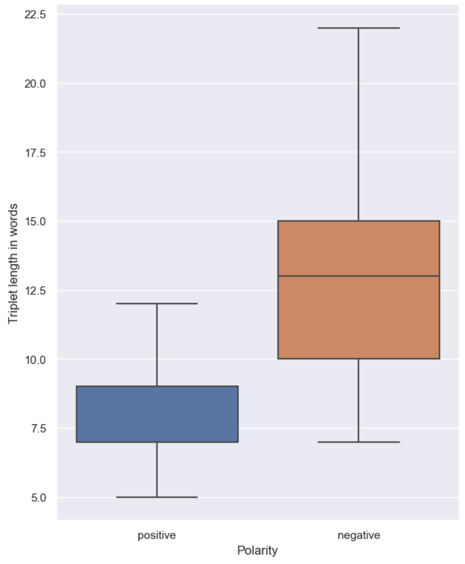


Figure 5.5: USMLE-Symp input size in number of components for the whole dataset.

whole dataset: again, this is relevant because the higher the number of components, the harder it is to correctly verbalize the input set without omitting important information or hallucinating details that are not really there. We notice that most input sets are short and contain only a maximum of three components, which may indicate that our data is easier to verbalize with respect to WebNLG. Indeed, despite the complexity of the medical topics, our triplets are pretty standardized as they are generated ad hoc from previously extracted information and not carefully selected from an existing knowledge base like DBpedia. Next, we investigate whether any correlation exists between the



(a) Category-wise input length in words.



(b) Polarity-wise input length in words.

Figure 5.6: Box plot of category-wise and polarity-wise input length in words. The plot refers to the whole USMLE-Symp dataset.

input length in words and its category/polarity. The results of such analysis are reported in Figure 5.6: we omit category *Connective tissue* from the analysis because it only has one triplet in the whole dataset - more precisely, in the test split - and as such, the displayed box plot would make no sense for it. We also do not show eventual outliers, which explains why the maximum number of words for a triplet in Figure 5.6a is around 22. We immediately notice that some categories (i.e., Cellular phenotype, Head and neck, Prenatal and Birth) tend to have longer inputs and thus can be assumed to be more complex to verbalize. Categories like *Respiratory System* and *Muscolature* instead tend to have smaller triplet sets, often not exceeding 10 words each. Moreover, Figure 5.6b shows that, while we do have fewer examples with negative polarity than with positive, the negative ones are often longer and have a bigger spread.

Finally, Table 5.5 shows some summary statistics about the length in both words and chars of the input and of the target descriptions. Because we pointed out the presence of outliers, we do not just show mean, minimum, maximum, and standard deviation but also the 99^{th} percentile. We can see that 99% of the input triplet sets have a length in words equal to or lower than 42, while for the target verbalizations the value becomes 26. We also notice that no triplet has a length lower than 5 and that no description in the training set has less than 6 words, with mean values that are respectively 10.07 ± 7.52 and 11.93 ± 3.86.

Table 5.5: USMLE summary statistics for both input and verbalizations.

	Type	**Mean**	**Min**	**Max**	**Std**	**99%**
Triplet	chars	75.59	33.00	976.00	56.74	342.44
	words	10.07	5.00	147.00	7.52	42.0
Verbalization	chars	76.37	29.00	322.00	28.56	205.00
	words	11.93	6.00	45.00	3.86	26.0

Chapter 6

Experiments

In this chapter, we describe in detail the experimental settings used during the training phase and the relative evaluation procedures. Section 6.1 gives an overview of the libraries and tools we adopted, while Section 6.2 focuses on the evaluation strategies used for the general purpose and for the medically-oriented portions of our work.

The remaining sections present an analysis of the results we obtained applying each of the three prompting techniques described in Chapter 4 to the WebNLG benchmark and to our own clinical data. In particular, Section 6.3 focuses on the general domain DBpedia triplets, while Section 6.4 details the performances obtained on the USMLE-Symp dataset.

6.1 Experimental Setup

The code developed during this thesis[1] is written in *Python*, a high-level programming language that has become a staple in the scientific domain. Indeed, Python allows for straightforward data manipulation and analysis, making it easy to conduct complex statistical evaluations, implement machine learning algorithms and create data visualizations. We now provide a non-exhaustive list of the libraries used for this work, focusing on the most relevant ones. We rely on *NumPy* for vector operations on CPU and on *Pandas* DataFrames to deal with Comma-Separated Values (CSV) files containing the evaluation results. Pandas is also employed to read and write *Pickle* files, that

[1]https://github.com/mwritescode/data2text-prompting

is, to serialize and de-serialize a Python object, converting it to a byte steam that can be saved on disk, allowing for easy reconstruction of the original object later on. This is especially important when dealing with the clinical USMLE-Symp data. Moreover, all the figures and plots we show in this work are realized through *Matplotlib* in conjunction with *Seaborn*, two data visualization packages that allow for effortless integration with NumPy and Pandas.

For implementing and training our neural architectures we rely on *PyTorch*, an open-source machine learning library providing an automatic differentiation system and tensor computation with strong acceleration via GPU. For code organization and model sharing we exploit *HuggingFace*, a data science platform providing tools to build, train and deploy deep learning models and applications. In particular, we build upon the *HuggingFace transformers* [52] library to develop our prompting techniques and rely on the online *HuggingFace hub* for access to pre-trained models' checkpoints. We also employ the Gensim library for accessing and using pre-trained vector embeddings like GloVe [31].

All our experiments and relative evaluations were either performed on the GPU cluster made available by the Department of Computer Science and Engineering of the University of Bologna, on Google Colaboratory or on Kaggle. The latter is an online community platform allowing machine learning practitioners to collaborate with other users, publish datasets and compete on data science challenges. When using this platform, which provides 30 hours per week of free GPU usage to its users, we always selected the NVIDIA Tesla P100 GPU, with 16 GB of RAM. Colab is an online platform that allows to execute Python code in the browser and which requires minimal setup. Their free tier allows users to exploit some computational resources free of charge while abiding by certain usage limits; when using this platform we were often assigned an NVIDIA Tesla T4 GPU with 16 GB of RAM. Finally, the university cluster is a High-Performance Computing (HPC) cluster composed of 10 nodes, each one equipped with an NVIDIA GeForce RTX 2080 Ti GPU and an Intel Xeon quad-core CPU. Computational resources on this cluster can be accessed by submitting a request to a *SLURM service*, an open-source and fault-tolerant job scheduling system.

The training runs that were executed through Colaboratory and Kaggle used *Jupyter* *notebooks*, as did all the results' analyses. Finally, experiment and weight tracking is realized through Weights & Biases (W&B)[2], a free online system for dataset and model versioning, as well as for experiments and metrics logging.

6.2 Evaluation Strategy

All our evaluations are performed on validation and test sets that vary between domains and which have been described in Chapter 5, when presenting the datasets. Following both Li and Liang [18] and Clive et al. [6], we evaluate our data-to-text task using BLEU [30], METEOR [3] and TER [45], as well as show some non cherry-picked examples for qualitative evaluation. In particular, for both WebNLG and USMLE-Symp we use BLEU as our main validation metric, which we periodically track over training. For this step, we make use of HuggingFace's *evaluate* library, which provides a straightforward, consistent, and reproducible way to evaluate natural language processing models.

Regarding the final scores on the test set, for WebNLG we use the official evaluation protocol[3], which assesses the quality of the generated verbalizations in terms of the three automatic metrics mentioned above and previously detailed in Section 3.3.2. The provided script produces a score for the whole dataset and also separate scores for seen and unseen categories, all of which are reported in the tables we propose in Section 6.3. We should note that this benchmark evaluation only uses at most three reference verbalizations per example, as it is known that variations in the number and quality of the references make the comparison of scores across datasets troublesome. This means that, if the triplet set i is associated with $K > 3$ ground truth descriptions, only the first three will be used in the computation of BLEU, METEOR, and TER. Using the official evaluation script allows for easy comparison with current state-of-the-art results on the WebNLG benchmark, which is especially important in the context of the new proposed architectures. Finally, the test scores for USMLE-Symp are again computed using the standardized implementation of BLEU, TER, and METEOR provided by

[2]The W&B project for this work is available at https://wandb.ai/mwritescode/data2text-prompting

[3]available at https://gitlab.com/webnlg/webnlg-automatic-evaluation

HuggingFace's evaluate library.

6.3 Experimental Results on WebNLG

For the WebNLG dataset, we experiment with the three model types described in Section 3.1. In particular, we employ two versions of T5 [38], `t5-base` and `t5-large`, respectively with 220M and 770M parameters. Moreover, following Li and Liang [18], we also experiment with two versions of GPT-2 [37], namely `gpt2-medium` (350M parameters) and `gpt2-large` (770M parameters). Regarding BART [16], we only use the more expressive `bart-large`, with 560M parameters. The following sections illustrate our results with each of the three prompting techniques described in Chapter 4, that is, Prefix Tuning [18], Control Prefixes [6] and Prefix Pooling.

The input triplets are linearized according to the scheme shown in Table 6.1. Notice that the linearization employed for T5 is different than the one we use for the other models, as preliminary experiments show it to be more effective. Moreover, we should point out that, differently from Clive et al. [6], we do not use three special tokens (`<H>`, `<R>`, and `<T>`) to indicate the start of subject, predicate, and object in a triplet. Indeed, one of the reasons why we choose prompting techniques is their ability to achieve good results while keeping the underlying pre-trained language model fixed; however, adding the aforementioned special tokens would mean having to train at least a portion of its embedding matrix.

Table 6.1: Triplet linearization scheme.

Triplet set	(subject$_1$, predicate$_1$, object$_1$), (subject$_2$, predicate$_2$, object$_2$)
GPT-2/BART	| subject$_1$: predicate$_1$: object$_1$ | subject$_2$: predicate$_2$: object$_2$
T5	subject$_1$ | predicate$_1$ | object$_1$ & subject$_2$ | predicate$_2$ | object$_2$

6.3.1 Prefix Tuning

As explained in Section 3.1, T5 is trained on a multi-task objective through the help of some textual prefixes. For instance, for a summarization task we could prepend *"summarize: "* to the actual input, while in order to translate between French and

English, we could use prefixes like *"translate French to English: "*. Following Clive et al. [6], we try to add *"translate Graph to English: "* to the beginning of all our input examples: this makes our data-to-text task more closely resemble the pre-training objective. The effects of using such a textual prefix are verified through preliminary experiments on both `t5-base` and `t5-large`. In particular, we train both models in standard Prefix Tuning setting for 7 epochs and employ the validation BLEU for early stopping. We use Adafactor [43] as our optimizer with a starting learning rate of 5e-7 and a linear decay scheduler with 2000 warm-up steps. We also keep the backbone completely frozen and do not update batch statistics nor use dropout for the underlying pre-trained language model. The verbalizations are generated using beam search decoding with 5 beams. From the graph in Figure 6.1 we can immediately see that the validation loss curve is noticeably lower when using the preamble, indicating that the similarity to the pre-training task has an impact on model performance. Despite this, Table 6.2 shows that the differences in BLEU scores on the validation set are not as evident. Moreover, the aforementioned results from the automatic evaluation procedure actually imply that the preamble harms performances instead of helping them.

Table 6.2: Validation BLEU for `t5-base` and `t5-large` with and without textual preamble (prefix tuning setting).

	Preamble	BLEU
`t5-base`	-	**65.35**
	✓	65.09
`t5-large`	-	**66.04**
	✓	65.86

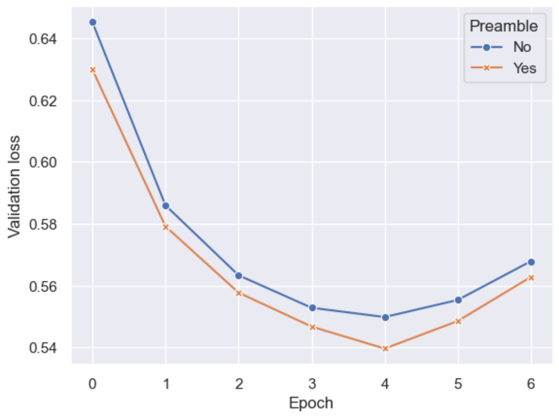


Figure 6.1: Validation loss for `t5-large` with and without textual preamble (prefix tuning setting).

Since these preliminary results contradict both Figure 6.1 and the hypothesis set forth by Clive et al. [6], we also perform a qualitative evaluation of the generated text. We notice that in 25% of the cases, the produced description remains identical when omitting the textual prompt, and show in Table 6.3 some examples of input triplets for which model `t5-large` generates two different verbalizations. As we can see, oftentimes when the preamble *"translate Graph to English: "* is included, the descriptions are more coherent and fluent, leading us to hypothesize that this is one case where automatic evaluation metrics may fail. We thus decide to use the textual prefix in all the following experiments, despite the slightly lower scores it achieves on the validation set. We also point out that both versions of `t5-large` seem to have learned to correctly verbalize the input triplet, despite errors or imprecisions in the label. For instance, in the second example of Table 6.3 both models state that the president/leader of the United States is John Roberts, which - while factually incorrect - is implied by the input triplets. The annotator who provided the reference description, instead, probably recognized that the triplet itself was factually incorrect and tried to resolve the error: in doing this, they introduced additional information (i.e., the fact that John Roberts is the Chief Justice of the United State), that the model has no way of knowing just by looking at the input. We note that this is a potential problem with crowdsourced datasets that should be further looked into in order to produce high-quality results: crowdsourced data is indeed more flexible than template-based verbalizations but it increases the risk of having incorrect gold standards.

We now experiment with the layer dependency introduced in Section 4.1 for the prefix parametrization. We train all our models for 7 epochs and perform early stopping on the validation BLEU. The T5 models all use the aforementioned textual preamble, while for a more detailed view of the model-wise training hyperparameters we refer to Appendix A. Table 6.4 shows the results in terms of validation BLEU for each of our five models, with and without layer dependency. We notice that this modification does not have a big impact on performance, as for all architectures the difference is below 1 BLEU point. In particular, despite what our preliminary experiments showed, we notice that only `t5-large` seems to benefit from the introduction of layer dependency, while for all other models, the modification actually appears to hurt performances. Figure 6.2 shows the learning curve in terms of validation BLEU for the model with the largest decrease in performance, that is, `gpt2-medium`: we notice that layer dependency seem to help the model achieve better generation capability at the start of

Table 6.3: WebNLG validation set, qualitative examples for model `t5-large`. The checkmark ✓indicates the use of the preamble *"translate Graph to English: "*.

Triplet	(Angola,_Indiana, isPartOf, Pleasant_Township,_Steuben_County,_Indiana)
References	1. Angola, Indiana is part of Pleasant Township, which is in Steuben County.
T5-large ✓	Angola is part of Pleasant Township, Steuben County, Indiana.
T5-large	Angola, Indiana is part of Pleasant Township, Steuben County, Indiana.
Triplet	(United_States, ethnicGroup, African_Americans), (United_States, language, English_language), (United_States, leaderTitle, President_of_the_United_States) (United_States, leaderName, John_Roberts), (250_Delaware_Avenue, location, United_States)
References	1. 250 Delaware Avenue is located in the United States, where the leader is known as the President of the United States, supported by the United States of America‘s Chief Justice, John Roberts. The main language is English, and one of the ethnic groups is the African Americans.
T5-large ✓	250 Delaware Avenue is located in the United States, where English is the language, African Americans are one of the ethnic groups and the leader is John Roberts.
T5-large	250 Delaware Avenue is located in the United States, where English is the language and African Americans are an ethnic group. The leader of the United States is called John Roberts and the leader is known as the President of the United States.
Triplet	(A_Long_Long_Way, language, English_language), (English_language, spokenIn, Great_Britain), (A_Long_Long_Way, followedBy, The_Secret_Scripture)
References	1. A Long Long Way (followed by the book The Secret Scripture) is written in the English language, which is spoken in Great Britain.
T5-large ✓	A Long Long Way was written in English, which is spoken in Great Britain. It was followed by The Secret Scripture.
T5-large	A Long Long Way was written in English which is the language spoken in Great Britain. It was followed by The Secret Scripture.
Triplet	(Acharya_Institute_of_Technology, officialSchoolColour, 'Blue, White and Orange'), (Acharya_Institute_of_Technology, was given the Ťechnical Campusśtatus by, All_India_Council_for_Technical_Education)
References	1. The official school colours for Acharya Institute of Technology are blue, white and orange and the school got its Technical Campus status from the All India Council.
T5-large ✓	The official school colours of the Acharya Institute of Technology are blue, white and orange and it was given the 'Technical Campus' status by the All India Council for Technical Education.
T5-large	The Acharya Institute of Technology was given the 'Technical Campus' status by the All India Council for Technical Education in the colours of blue, white and orange.
Triplet	(Adare_Manor, completionDate, 1862), (Adare_Manor, architect, James_Pain), (Adare_Manor; owner; J._P._McManus)
References	1. James Pain was the architect of Adare Manor which was completed in 1862 and is owned by J P McManus.
T5-large ✓	James Pain was the architect of Adare Manor which was completed in 1862 and is owned by JP McManus.
T5-large	James Pain designed the Adare Manor which was completed in 1862 and owned by J.P.McManus.

training, but then rapidly saturates. We speculate that the alternative parametrization could indeed be an asset when dealing with a limited computational budget.

Table 6.4: BLEU scores on WebNLG validation set for all our prefix tuning models, with and without layer dependency

	Layer Dep	BLEU
t5-base	-	**65.09**
	✓	64.85
t5-large	-	65.86
	✓	**66.16**
gpt2-medium	-	**66.19**
	✓	65.23
gpt2-large	-	**66.06**
	✓	66.00
bart-large	-	**64.92**
	✓	64.88

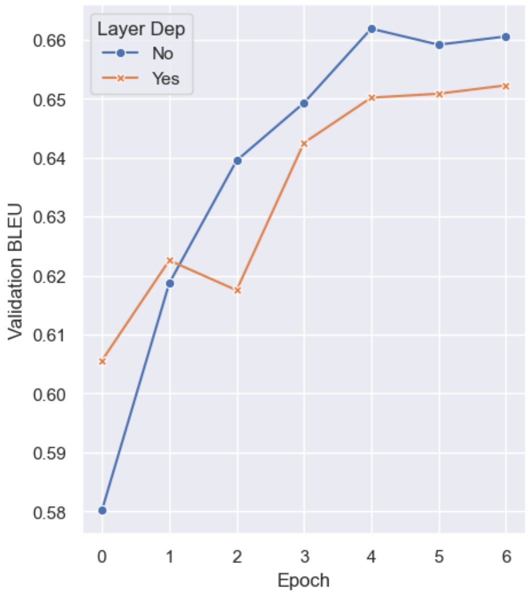


Figure 6.2: Validation BLEU for `gpt2-medium` in prefix tuning setting with and without layer dependency.

Finally, Tables 6.5 and 6.6 show the results obtained by our best models on the test set, evaluated using the official script. Because the decoding strategy can play an important role when evaluating a generative model, we show the performances for the two schemes described in Section 3.3.1. That is, we follow the current state-of-the-art results in data-to-text [6, 18] and use beam search with 5 beams, but we also experiment with contrastive search, a recent strategy proposed to improve open-ended text generation without incurring in repetitions or losing semantic meaning. In particular, since we are performing conditional text generation, we set the degeneration penalty to a low value of 0.1 in order not to penalize too much the repetition of words and n-grams, which is often welcome, if not required, in our setting. Note that, in both tables, a checkmark (✓) next to a model name indicates that we are using prefix parametrization with layer dependency. We can immediately notice that beam search decoding decidedly outperforms contrastive search for our setting. The BLEU score is particularly affected by this change, dropping by about 2 points, while METEOR and TER only vary by 0.01/0.02. Because no improvements are seen, in any of the three scores, when using contrastive search decoding, the following experiments will only show the results achieved by beam search. We also notice that the model `t5-large` with layer dependency obtains the best scores across all metrics on the test set: it is interesting to point out that this was only our second-best model on the validation set, as it was slightly outperformed by `gpt2-medium`. The latter model has instead dramatically lower performances on the test set, especially on the unseen partition: this indicates that `gpt2-medium` finds it harder to generalize to new domains. Appendix B shows some qualitative examples of the verbalizations generated by our models and provides a brief analysis of the differences.

Table 6.5: Prefix tuning test results on WebNLG using the beam search decoding strategy. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑			**METEOR ↑**			**TER ↓**		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
`t5-base`	64.14	46.88	56.46	**0.46**	0.39	0.43	**0.33**	0.48	0.40
`t5-large` ✓	**65.12**	**50.60**	**58.58**	**0.46**	**0.41**	**0.44**	**0.33**	**0.43**	**0.37**
`gpt2-medium`	61.63	43.77	53.59	0.44	0.36	0.40	0.35	0.49	0.41
`gpt2-large`	63.83	45.99	55.78	0.45	0.38	0.42	**0.33**	0.49	0.40
`bart-large`	63.88	48.50	56.96	0.45	0.40	0.43	0.34	0.45	0.39

Table 6.6: Prefix tuning test results on WebNLG using the contrastive search decoding strategy. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑			**METEOR ↑**			**TER ↓**		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
`t5-base`	62.66	43.70	53.90	0.45	0.38	0.42	0.35	0.50	0.42
`t5-large` ✓	**63.86**	**49.18**	**57.24**	**0.46**	**0.40**	**0.43**	**0.34**	**0.44**	**0.39**
`gpt2-medium`	60.67	39.08	50.71	0.44	0.36	0.40	0.37	0.59	0.47
`gpt2-large`	63.41	40.96	52.84	0.45	0.38	0.41	0.35	0.56	0.45
`bart-large`	61.48	45.16	54.16	0.44	0.38	0.41	0.35	0.47	0.41

6.3.2 Control Prefixes

Given the non-conclusive results regarding the effectiveness of layer dependency for the prefix parametrization, in the control prefixes setting we realize our experiments both with and without it. We still use all the five model types mentioned in Section 6.3, that is, T5 both in sizes `base` and `large`, GPT-2 `medium` and `large`, and BART `large`. All our pre-trained models have between 220M and 770M parameters.

We use 10 control prefixes at training time, corresponding to the ten seen DBpedia categories. At test time, we follow Clive et al. [6] and deal with unseen categories by performing zero-shot learning: we map each new class to the closest seen category in the embedding space, where the embeddings are computed using GloVe [31]. We train for a maximum of 7 epochs and perform early stopping based on the validation BLEU; the model-wise hyperparameters we use are reported in Appendix A. Table 6.7 shows our models' results in terms of validation BLEU both with and without layer dependency.

Table 6.7: Comparison of validation BLEU scores on WebNLG for all control prefixes models, with and without layer dependency

	Layer Dep	**BLEU**
`t5-base`	-	65.88
	✓	**65.92**
`t5-large`	-	**66.11**
	✓	66.06
`gpt2-medium`	-	65.77
	✓	**66.03**
`gpt2-large`	-	**65.98**
	✓	65.86
`bart-large`	-	**65.28**
	✓	65.26

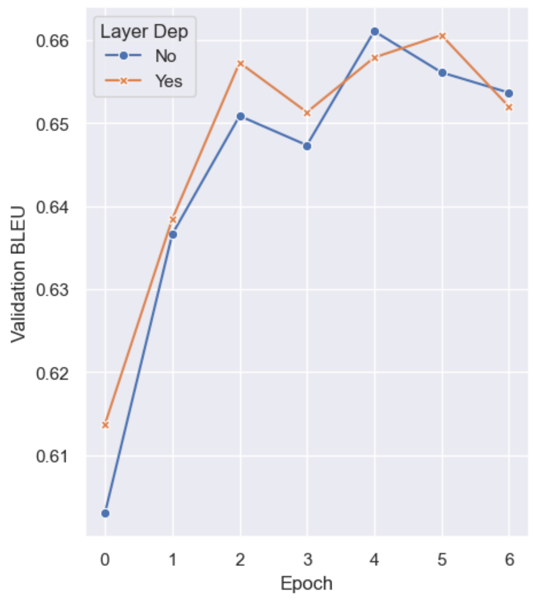


Figure 6.3: Validation BLEU for `t5-large` in control prefixes setting with and without layer dependency.

We immediately notice that layer dependency seems to improve performances for the smaller pre-trained language models, that is, `t5-base` and `gpt2-medium`, not have much impact at all on our middle-sized model (i.e., `bart-large`) and cause a slight capability degradation in the two remaining bigger architectures. This seems to indicate that the prior is particularly effective to boost the performances of smaller models. We also hypothesize that our naive addition to the prefix's parametrization works slightly better in this setting because we apply it after the concatenation of the two prefix components, thus somehow merging static and dynamic prompts.

Figure 6.3 shows the learning curve in terms of validation BLEU for `t5-large`, which - as we just pointed out - appears to suffer slightly from the change in parametrization strategy: we can see that the BLEU scores for the first epochs are decidedly higher when using layer dependency, but then rapidly saturate. This seems to confirm our previous hypothesis that the modification is an asset particularly when training on a limited computational budget. Finally, Table 6.8 shows the results of the official

Table 6.8: Control prefixes test results on WebNLG using the beam search decoding strategy. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑			**METEOR ↑**			**TER ↓**		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
`t5-base` ✓	64.01	45.97	55.93	**0.46**	0.39	0.42	0.34	0.48	0.41
`t5-large`	**64.94**	**49.61**	**58.05**	**0.46**	**0.40**	**0.43**	**0.33**	**0.44**	**0.38**
`gpt2-medium` ✓	62.26	41.01	52.77	0.44	0.36	0.40	0.35	0.52	0.43
`gpt2-large`	63.51	42.97	54.36	0.45	0.37	0.41	0.34	0.50	0.42
`bart-large`	63.51	45.00	55.25	0.45	0.38	0.42	0.34	0.47	0.40

WebNLG evaluation script on the test set for our best configuration of each model type. Notice that a checkmark (✓) next to the model name indicates that we are using layer dependency, as described in Chapter 4. We can see that `t5-large` achieves the overall best performances on all three metrics; this is especially true for what concerns the unseen categories, where it obtains approximately 4 BLEU points more than the next best model (i.e., `t5-base`). We should also point out that `gpt2-medium` achieves very similar performances on the validation set, but generalizes poorly to new examples that cover different topics. This leads us to assert that `t5-large` is able to better adapt to unfamiliar domains.

It is also interesting to note that, despite having the smallest amount of parameters, `t5-base` with layer dependency performs better than much bigger pre-trained models like `gpt2-large` and `bart-large` on all three metrics, both in terms of seen and unseen categories. This is not necessarily surprising if we consider that textual prefixes were already used in T5's pre-training objective, meaning that this class of models is especially well-suited to work with prompting/prefix tuning techniques. For some qualitative examples of the verbalizations generated using control prefixes, we refer the reader to Appendix B, where we also provide a brief analysis of the differences with respect to standard prefix tuning.

6.3.3 Prefix Pooling

In order to further analyze the eventual benefits provided by layer dependency, we realize our experiments in the Prefix Pooling setting both with and without changing the parametrization. We still use the five model types employed for Prefix Tuning and Control Prefixes, all with sizes between 220 and 770 million parameters.

We use a pool of 10 prefixes, each one with length two; for every example in our dataset we select the $k = 3$ prefixes with key most similar to the query produced from the input embeddings. We also limit the size of the static task prefix to three in order to curb the risk of overfitting. We train for a maximum of 7 epochs and again perform early stopping based on the validation BLEU; for more detailed, model-wise hyperparameters, we refer to Appendix A.

Table 6.9: Comparison of validation BLEU scores on WebNLG for all prefix pooling models, with and without layer dependency

	Layer Dep	BLEU
`t5-base`	-	**64.76**
	✓	64.63
`t5-large`	-	**65.74**
	✓	65.46
`gpt2-medium`	-	**65.94**
	✓	65.91
`gpt2-large`	-	**65.94**
	✓	65.87
`bart-large`	-	**65.45**
	✓	65.01

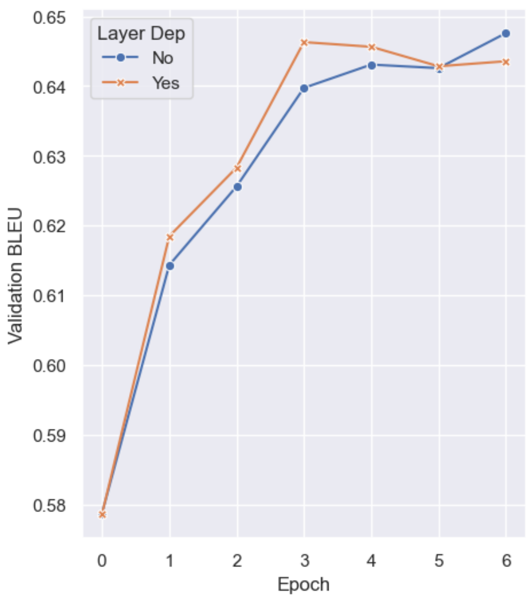


Figure 6.4: Validation BLEU for `t5-base` in prefix pooling setting with and without layer dependency.

Table 6.9 shows the performances achieved by our models with and without layer dependency: we notice that, when the dynamic prefix is chosen based on similarity with the input embeddings, the prior about layer dependency does not produce improvements on most models' overall performances. This might be due to the fact that we are already pushing the expressivity of our architecture with the input-dependent prefixes, so that adding another element of complexity simply causes degradation. Figure 6.4 shows how the validation BLEU score changes during training for model `t5-base` when varying the parametrization: we can, once again, observe that our modification helps to achieve a better score within a limited amount of epochs before performances saturate. Finally, Table 6.10 shows the results that our best configurations achieved on the test set according to the official evaluation script.

Table 6.10: Prefix Pooling test results on WebNLG using the beam search decoding strategy. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑			**METEOR ↑**			**TER ↓**		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
`t5-base`	**64.50**	47.50	56.86	**0.46**	0.39	0.43	**0.33**	0.46	0.39
`t5-large`	64.49	**51.32**	**58.52**	**0.46**	**0.41**	**0.44**	**0.33**	0.44	**0.38**
`gpt2-medium`	62.76	43.40	54.06	0.45	0.37	0.41	0.34	0.49	0.41
`gpt2-large`	63.81	46.85	56.13	0.45	0.38	0.42	**0.33**	0.50	0.41
`bart-large`	63.14	50.45	57.44	0.45	0.40	0.42	**0.33**	**0.43**	**0.38**

We can immediately see that `t5-large` is the best-performing model, closely followed by `t5-base`. We however also observe how, with the new dynamic prompting technique we propose, `bart-large` obtains the same TER scores, and gets quite similar BLEU and METEOR values, especially for what concerns the unseen categories. This is promising, as `bart-large` is much smaller than `t5-large`, containing approximately 32% fewer parameters. Moreover, comparing with Tables 6.5 and 6.8, we notice that Prefix Pooling improves performances on unseen categories for several models (i.e.: `bart-large`, the two GPT-2 and `t5-large`) over both standard Prefix Tuning and Control Prefixes, while achieving prompt dynamicity. Thus, the new technique might be better equipped to adapt to changes in topic or domain of the triplets.

Finally, notice that Prefix Pooling lets us design prompts that vary for every input example, thus reflecting its complexity, without needing any external conditional information and without compromising on performances with respect to previous state-of-the-art approaches.

6.4 Experimental Results on USMLE-Symp

For our medical USMLE-Symp dataset, we again experiment with the three generative models described in Section 3.1, but now also consider their versions pre-trained specifically on clinical data. In particular, in Section 6.3 we proved that `t5-large` always outperforms the smaller model `t5-base`, which is hardly surprising as the more parameters our model has the more it is expressive. However, we note that the USMLE-Symp dataset, while targeting a more complex domain, should overall be considered easier to solve with respect to WebNLG as it is not crowdsourced or expertly annotated but mostly template-based. We thus acknowledge that more expressive models like `t5-large` might overfit our second dataset and do not limit our analysis to it, but still consider also `t5-base`. For both models, we also employ their clinical counterparts, which are called `SciFive-large` and `SciFive-base`.

We then note that Microsoft provides only one version of BioGPT [22], whose parameter count roughly corresponds to that of `gpt2-medium`. We thus focus our experiments only on these two models, discarding the more expressive `gpt2-large` for the medical domain. Finally, we consider both `bart-large` and the corresponding `biobart-large`, which is continually pre-trained using PubMed's abstracts. Also notice that, since this dataset is smaller than WebNLG, we set the maximum amount of epochs to 4 in order to limit overfitting and perform early stopping on the validation BLEU when needed. We train all our models with layer dependency, as our previous experiments showed its capacity to boost performances when working with a limited computational budget.

The input triplets are linearized according to the same scheme used for WebNLG, which is shown in Table 6.1. We again point out that, differently from Clive et al. [6], we do not employ any special tokens to signal the start of subject, predicate, and object in a triplet. The following sections illustrate the results we obtain with each of the prompting techniques described in Chapter 4. In particular, our tables refer to the same three automated metrics we used for WebNLG, that is, BLEU [30], METEOR [3] and TER [45]. We use the stable and reproducible implementation provided by Hugging-Face's evaluate library and limit the maximum number of references per input example to two. We should note that, with this implementation, TER can take on any value greater than zero, which is the perfect score and indicates that no edits are necessary to turn the predicted verbalizations into their corresponding references.

6.4.1 Prefix Tuning

Here we experiment with all four models mentioned above (i.e.: `gpt2-medium`, `t5-base`, `t5-large`, and `bart-large`) and compare the results with their medical versions, that were pre-trained, from scratch or continually, on clinical data. Following the results obtained for WebNLG in Section 6.3, we pre-pend the textual preamble *"translate Graph to English: "* to all T5/SciFive models, in order to bring our task closer to those on which the models were pre-trained. As previously stated, we train for 4 epochs and use the validation BLEU score for early stopping; following both Li and Liang [18] and Clive et al [6], we also keep the underlying PLM frozen and only update the prefix parameters. The model-wise hyperparameters we use are detailed in Appendix A, together with the parameters used for generation. Again, following our previous results on the comparison between contrastive and beam search, we focus on the latter as it guarantees higher-quality generations, especially for what concerns the BLEU score.

Table 6.11 shows the performances of our models and of their medical counterparts in terms of validation error: as we can see model `bioGPT` achieves the overall best results, with `SciFive-base` coming in as a close second. Moreover, as expected, the models pre-trained on clinical data coming from PubMed's abstracts and full-text articles generally perform better in terms of automatic scores. We however already noted how the automated scores do not always have a strong correlation with human evaluation, as they typically focus on aspects like edit distance and n-grams overlap and not as much on semantics. We thus report some qualitative examples over the

USMLE-Symp validation set, in order to better assess the difference made by domain-specific information. This is especially interesting for the data-to-text task because, to join together two or more triplets in a coherent and fluent way, the most important capability a model should have is grammatical/lexical knowledge of the English language. Information about the correctness of the underlying medical data is not a strict requirement for this task because the models should assume the information provided in the triplets to be correct and simply verbalize it, without fixing eventual errors in the input. We thus argue that knowledge of the English language is more important than comprehension of the domain-specific topic alone; however, the generative pre-trained language models have been deliberately designed to maintain a high degree of linguistic knowledge and the added medical information can make it easier to describe inputs full of otherwise unfamiliar medical terms. Table 6.12 displays the generated

Table 6.11: Comparison of validation BLEU scores on USMLE-Symp for general-purpose and domain-specific models. The scores refer to the Prefix Tuning setting.

	Domain Specific	**BLEU**
`t5-base`	-	97.91
	✓(`SciFive-base`)	**98.46**
`t5-large`	-	**97.90**
	✓(`SciFive-large`)	97.88
`gpt2-medium`	-	98.01
	✓(`bioGPT`)	**98.49**
`bart-large`	-	98.21
	✓(`biobart-large`)	**98.23**

verbalizations for an instance with negative polarity coming from the validation set, while we refer the reader to Appendix B.2 for more qualitative examples and a deeper analysis of the differences between models. From Table 6.12 we can immediately notice how `gpt2-medium` produces a somewhat non-sensical explanation with wrong polarity (i.e.: *positive*), while the domain-specific `bioGPT` produces an almost perfect description. It only mistakenly includes *pain* amongst the symptoms of Tuberculosis that the patient does not have, probably misinterpreting the triplet `(patient, Has_symptom, pain)`. Something similar also happens for models `t5-base` and `SciFive-base`, where the general-purpose PLM gets the polarity wrong and the domain-specific model adds symptoms the patient is actually suffering from in the list of reasons why the diagnosis is incorrect. Differently, our two bigger models (`t5-large` and `bart-large`) generate problematic outputs even when switching to the corresponding clinically oriented architectures: indeed, BART's output contains all the correct symptoms and has the correct polarity, but it fails to identify a causality link between those symptoms and the lack of a diagnosis. Similarly, BioBART, contains the correct symptomatology but produces an explanation with opposite polarity and misses the causality link. The same can be said

Table 6.12: Qualitative generation example from the USMLE-Symp validation set. The example is of category *Immunology* and has *negative* polarity

Triplet	(Tuberculosis, Has_as_symptom, Fever), (Tuberculosis, Has_as_symptom, Abnormal lung morphology), (Tuberculosis, Has_as_symptom, Cough), (Tuberculosis, Has_as_symptom, Weight loss), (Tuberculosis, Has_as_symptom, Fatigue), (patient, is_a, 37-year-old woman), (patient, Does_not_suffer_from, Tuberculosis), (patient, Has_symptom, shortness of breath), (patient, Has_symptom, difficulty walking), (patient, Has_symptom, pain), (patient, Has_symptom, aches),(patient, Has_symptom, pains)
Reference	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.
gpt2-medium	These symptoms indicate that the patient my suffer from Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and People with the following symptoms: Shortness of breath, Aches and Pain.
bioGPT	The following list of symptoms is not found in the case description, therefore the Tuberculosis diagnosis must be rejected: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and Pain.
t5-base	The patient, a 37-year-old woman, has been diagnosed with Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
SciFive-base	These symptoms indicate that the patient does not suffer from Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue, Pain and pain.
t5-large	These symptoms indicate that the patient has Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
SciFive-large	These symptoms indicate that the patient my suffer from Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and is_a 37-year-old woman.
bart-large	The patient, a 37-year-old woman, does not suffer from Tuberculosis. The patient has the following symptoms: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
biobart-large	TheThe patient is a 37-year-old woman and is affected by Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.

for `t5-large` and `SciFive-large`, as they both produce a positive explanation. This leads us to conclude that, given the relative simplicity of the dataset, the bigger models soon start to overfit the training data, thus losing the ability to correctly verbalize new samples. We also hypothesize that, in these situations, using the domain-specific model might hurt performances even more because it is naturally more expressive, having been trained to retain linguistic knowledge but also absorb medical information. In the following, we will not consider the `large` version of model T5 anymore for our experiments, as it does not produce any real performance improvement with respect to `t5-small`, which has 28% fewer parameters.

In Table 6.13 we now show BLEU, METEOR, and TER scores obtained by our three best models in Prefix Tuning setting on the test set. As we can see, despite not obtaining the best results in terms of BLEU on the validation set, BioBART performs the best on the test data, particularly for what concerns the TER score. This implies that, in general, a smaller amount of insertions, deletions, or substitutions are needed to recreate the reference text starting from the verbalizations it produces.

Table 6.13: Prefix Tuning test results on USMLE-Symp dataset using the beam search decoding strategy.↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑	**METEOR ↑**	**TER ↓**
`bioGPT`	96.26	0.982	3.46
`SciFive-base`	96.32	0.984	3.08
`biobart-large`	**96.78**	**0.986**	**2.78**

6.4.2 Control Prefixes

We previously showed that, in general, domain-specific models provide a small boost in performance for what concerns automated metrics and also help in maintaining the polarity consistent between reference and generation. We thus focus our attention on the more expressive medical models and try to see whether their performances could be further improved by introducing dynamic prefixes. We also showed that both `t5-large` and `SciFive-large` do not provide any improvements in terms of validation BLEU with respect to their `small` counterparts: because of this, our efforts now focus on the remaining three models, that is, `SciFive-base`, `biobart-large` and `bioGPT`. As before, we train for 4 epochs at most while keeping the underlying language model frozen and performing early stopping on the validation BLEU. We also reduce the length of the task-specific prefix to three, to account for the fact that this dataset is small and template-based and, therefore, easier than WebNLG. More detailed model-wise hyperparameters are described in Appendix A.

Polarity as guidance signal We noted that Prefix Tuning models struggle to assign the correct polarity to the produced verbalization, often generating "why" explanations when we require a "why-no" justification and vice-versa. We thus employ Control Prefixes [6] to specify the desired polarity of the output text and to steer the generation towards being of the correct type. In particular, the idea is to use the information about whether a diagnosis is positive or negative as a guidance signal in order to select the appropriate dynamic prefix, which is then concatenated to the task-specific prompt and should help exercise a more fine-grained control on the produced text. Table 6.14 shows the results obtained by our three models on the validation set: as we can see all architectures improved their BLEU score with respect to the Prefix Tuning setting and, particularly, `biobart-large`'s score increased by 0.42 points. Because the models'

Table 6.14: Validation BLEU scores for the USMLE-Symp dataset in Control Prefixes setting. These values were obtained when using *polarity* as a guidance signal.

	`BioGPT`	`SciFive-base`	`biobart-large`
BLEU	98.64	**98.66**	98.65

performances are all very close together, we also qualitatively examine some instances from the validation set, in order to verify whether the guidance signal has the desired effect. Table 6.14 shows the verbalization generated by each model for our running *Tuberculosis* example, while Appendix B.2 reports more examples for input triplets with positive polarity. We immediately see that both `bioGPT` and `biobart-large` produce extremely good verbalizations, where the polarity is correct, the causal relationship between symptoms occurrence and diagnosis is pointed out and no symptom is hallucinated. Model `SciFive-base` instead keeps considering *pain* as something the patient

Table 6.15: Qualitative generation example from the USMLE-Symp validation set. The texts have been generated using Control Prefixes with *polarity* as the guidance signal.

Triplet	(Tuberculosis, Has_as_symptom, Fever), (Tuberculosis, Has_as_symptom, Abnormal lung morphology), (Tuberculosis, Has_as_symptom, Cough), ... (patient, Has_symptom, aches),(patient, Has_symptom, pains)
Reference	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.
bioGPT	Tuberculosis has been rejected as a possible diagnosis for the patient due to the lack of the following symptoms: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
SciFive-base	The patient is not exhibiting these symptoms: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and Pain
biobart-large	The following list of symptoms is not found in the case description, hence the Tuberculosis diagnosis must be rejected: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.

is not experiencing and, what is more important, does not conclude anything about the diagnosis. This is particularly significant if we remember that SciFive obtains the best BLEU score on the validation set because it highlights how automatic evaluations of generative systems may fail. Notice that, despite the missing conclusions, we can still state that SciFive generates a verbalization with “negative” polarity, as it clearly captures that we are talking about missing symptoms.

Overall, we can then state that including polarity as a guidance signal has a positive impact on performance and helps steer the generation toward the desired type.

Category as a guidance signal Next, we try to use the HPO category associated with each triplet as the conditional information on which our dynamic prefixes are based. We have 20 general categories coming from the medical domain and only one unseen category present in the test set, that is, *Connective tissue*. We follow the zero-shot learning procedure described by Clive et al. [6] in order to deal with it: we assume semantic similarities between classes, compute their GloVe embeddings and then re-map the examples with unseen category to their closes seen class in the embedding space. In this case, *Connective tissue* gets remapped to *Skeletal system*. Table 6.16 displays the results obtained by our three models in terms of validation BLEU: as we can see, `SciFive-base` obtains once again the greatest score, closely followed by `bioGPT`. In

Table 6.16: Validation BLEU scores for the USMLE-Symp dataset in Control Prefixes setting. These values were obtained when using the *category* as a guidance signal.

	`BioGPT`	`SciFive-base`	`biobart-large`
BLEU	98.47	**98.49**	97.81

contrast, `biobart-large` performances seem to deteriorate with respect to the static Prefix Tuning setting. We also notice that these results are all significantly lower than those we got when using polarity as a guidance signal, suggesting that the input triplets composition does not vary a lot with its category. We nevertheless show in Table 6.17 the generated verbalizations for our running *Tuberculosis* example; we again refer the reader to Appendix B.2 for other description samples. We can immediately notice that the

Table 6.17: Qualitative generation example from the USMLE-Symp validation set. The texts have been generated using Control Prefixes with *category* as the guidance signal.

Triplet	(Tuberculosis, Has_as_symptom, Fever), (Tuberculosis, Has_as_symptom, Abnormal lung morphology), (Tuberculosis, Has_as_symptom, Cough), ... (patient, Has_symptom, aches),(patient, Has_symptom, pains)
Reference	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.
bioGPT	The disease Tuberculosis is typically characterized by these symptoms: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
SciFive-base	The 37-year-old woman is not affected by Tuberculosis.
biobart-large	These symptoms indicate that the patient my suffer from Tuberculosis: Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue. The patient is also a 37-year-old woman.

quality of the generation degraded: for instance, the text generated by `SciFive-base` only states that the patient is not affected by Tuberculosis, without citing any symptom at all. By contrast, `bioGPT` correctly captures the symptomatology associated with Tuberculosis but completely ignores the patient. Finally, `biobart-large` produces a complete justification but mistakes the polarity: we thus have a description of why the patient might have Tuberculosis instead of a “why-no” explanation. These results lead us to conclude that category information is not necessarily a useful addition when performing data-to-text, especially in situations where the dataset is small and all triplet sets have a similar structure and same predicates, independently from the HPO category of the disease they describe.

Category and polarity as guidance signals Despite having found that category information on its own does not help performances on the USMLE-Symp dataset, we still experiment with using the two guidance signals together. When doing this we effectively compute two separate sets of control prefixes, then select the appropriate one given the input's polarity from the first set and concatenate it to the left of the category-based dynamic prefix. Everything is then pre-pended as a left context to the actual model input. We speculate that letting polarity prefixes influence the category-based ones could make it easier for the model to work out interesting correlations between these two variables in our dataset, thus improving performances. Table 6.18 displays the results we obtain from this experiment in terms of validation BLEU: as we can see, `bioGPT` achieves the higher score we have registered up until now, while performances for `biobart-large` notably improve from the category-only setting, but remain below the obtained scores for both polarity-only Control Prefixes and Prefix Tuning. The results for `SciFive-large` instead drop lower with respect to both the other versions of Control Prefixes. However, we showed multiple times how simply using automatic

Table 6.18: Validation BLEU scores for the USMLE-Symp dataset in Control Prefixes setting. These values were obtained when using both *category* and *polarity* as a guidance signal.

	`BioGPT`	`SciFive-base`	`biobart-large`
BLEU	**98.81**	98.42	98.04

evaluation scores to evaluate a generative language model might be misleading. We thus display in Table 6.19 the verbalizations generated for our running example on *Tuberculosis* diagnosis, while more qualitative samples can be found in Appendix B.2. We immediately notice that, despite the better validation BLEU, the generated descriptions are not correct and pretty closely resemble those we obtained when just using the category as a guidance signal. Indeed, both `bioGPT` and `biobart-large` capture the lack of certain relevant symptoms in our patient, but do not draw any conclusions from it. The verbalization produced by `SciFive-large` instead includes the disease but mistakenly

Table 6.19: Qualitative generation example from the USMLE-Symp validation set. The texts have been generated using Control Prefixes with *category* and *poalrity* as the guidance signal.

Triplet	(Tuberculosis, Has_as_symptom, Fever), (Tuberculosis, Has_as_symptom, Abnormal lung morphology), (Tuberculosis, Has_as_symptom, Cough), ... (patient, Has_symptom, aches),(patient, Has_symptom, pains)
Reference	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.
bioGPT	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.
SciFive-base	The disease Tuberculosis is not affected by these symptoms: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and Pains.
biobart-large	The patient is a 37-year-old woman and is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue.

states that *Tuberculosis* is not "affected" by the relevant symptoms.

We thus conclude that, even when combined with polarity, information about the HPO category associated with a certain disease/symptom only causes the performances to worsen. This might be due to overfitting, as the dataset is quite small, or it might be due to the fact that our input triplet sets are quite standardized: they often have the same structure and use the same predicates, independently from the category. Thus, adding the categorical information might be completely irrelevant and just confuse our model. Given these observations, we conclude that the models produced with polarity-only Control Prefixes are the best so far and show in Table 6.20 their performances in terms of BLEU, METEOR, and TER on the test set. We notice that `biobart-large`

Table 6.20: Polarity-only Control Prefixes test results on USMLE-Symp dataset. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑	**METEOR ↑**	**TER ↓**
`bioGPT`	**97.63**	**0.989**	**2.39**
`SciFive-base`	97.35	0.987	2.87
`biobart-large`	96.08	0.981	3.45

is the only model for which performances on the test set slightly deteriorated with respect to standard Prefix Tuning; the other two models obtain an increase in BLEU of approximately one point each and a considerable decrease in TER, meaning that a smaller number of edits are necessary to turn the generated text into the reference verbalization.

6.4.3 Prefix Polling

We again focus our attention on the three domain-specific models that achieved the best results in Prefix Tuning and Control Prefixes, that is, `SciFive-base`, `bioGPT` and `biobart-large`, in order to verify whether the same results could be obtained using dynamic prefixes that do not rely on any additional information. As always, we train for a maximum of 4 epochs and perform early stopping on the validation BLEU score; we keep the underlying language model frozen and only train the prompt parameters. A more detailed account of model-wise parameters is shown in Appendix A. We use a pool of 10 prefixes from which the dynamic prompts are extracted based on similarity with the input embeddings, but we never use this dynamic prompt alone; instead, it is combined with a static, task-specific prefix.

Table 6.21 shows the results achieved by our models in terms of validation BLEU: we should note that `biobart-large` outperforms both `SciFive-base` and `bioGPT`, while also obtaining higher scores with respect to its Control Prefixes and Prefix Tuning variations. This seems to indicate that Prefix Pooling has a better generalization capability with respect to current state-of-the-art. Moreover, Table 6.22 shows the texts

Table 6.21: Validation BLEU scores for the USMLE-Symp dataset in Prefix Pooling setting.

	`BioGPT`	`SciFive-base`	`biobart-large`
BLEU	98.34	98.26	**98.68**

generated by all three models for our running *Tuberculosis* example, letting us examine the verbalizations from a qualitative point of view. As previously pointed out, Appendix B.2 contains more in-depth examples to individually assess the performances of this technique. In accordance with the validation BLEU results, we notice that the text generated by `biobart-large` has a high quality: it includes all the relevant

Table 6.22: Qualitative generation example from the USMLE-Symp validation set. The texts have been generated using Prefix Pooling.

Triplet	(Tuberculosis, Has_as_symptom, Fever), (Tuberculosis, Has_as_symptom, Abnormal lung morphology), (Tuberculosis, Has_as_symptom, Cough), ... (patient, Has_symptom, aches),(patient, Has_symptom, pains)
Reference	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.
bioGPT	The patient, a 37-year-old woman, has the following symptoms: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue and Pain.
SciFive-base	The disease Tuberculosis is typically characterized by these symptoms: Fever, Abnormal lung morphology, Cough, Weight loss, Fatigue, Pain and pains.
biobart-large	The patient is not exhibiting Fever, Abnormal lung morphology, Cough, Weight loss and Fatigue, hence the Tuberculosis diagnosis must be rejected.

symptoms that are missing from the case description, does not hallucinate information, and correctly concludes that the diagnosis should be refused. This is not the case for the remaining two models, as they both omit the conclusions and mistake the polarity for positive, despite also having good BLEU scores on the validation set.

Finally, Table 6.23 shows our models' results on the test set: as always, we show BLEU, METEOR, and TER computed using HuggingFace's evaluate library and with a maximum number of references equal to two.

Table 6.23: Prefix Pooling test results on USMLE-Symp dataset. ↑ indicates that the higher the metric the better, while ↓ indicates that we want to obtain the lowest possible score.

	BLEU ↑	**METEOR ↑**	**TER ↓**
`bioGPT`	**96.92**	**0.985**	**2.98**
`SciFive-base`	96.39	0.982	3.54
`biobart-large`	96.37	0.980	3.59

We see that despite what Tables 6.21 and 6.22 indicate, `bioGPT` obtains higher BLEU and METEOR scores than `biobart-large` on the test set, as well as a lower TER. This implies that `bioGPT` trained with Prefix Pooling generalizes better to unseen data with respect to the other models. Moreover, its scores also outperform the model's results in the Prefix Tuning setting, indicating that Prefix Pooling can achieve results on par with previous state-of-the-art while combining static and dynamic prefixes and without the need for any guidance signal. We again point out how completely flexible this approach is: it can be used in conjunction with a static task-specific prompt as we are doing right now, but also combined with Control Prefixes to steer the generated text toward having the desired characteristics. Finally, it could be used alone in a completely dynamic setting.

Chapter 7

Discussion

The main contribution of this thesis is two-fold: we first focused on analyzing and improving the weak points of existing prompting techniques for natural language generation, particularly concentrating on the data-to-text task. Next, we applied the aforementioned techniques to the medical domain, with the express aim to verbalize triplet sets containing all the necessary information for explaining a positive or negative diagnosis, given the clinical case. In this way we realize discursive justifications, moving away from the single template-based explanations that are produced by Marro et al.'s system [23]. We now summarize the primary results obtained by the experiments described in Chapter 6: Section 7.1 focuses on our modifications to the Prefix Tuning architecture and highlights the impact of the decoding strategy on the results of a generative model. Section 7.2 instead focuses on the performances obtained on the medical data, on highlighting the advantages of using a domain-specific model, and on the limitations of a data-to-text dataset where descriptions have been produced starting from template verbalizations.

7.1 WebNLG: Analysis of Results

We use the WebNLG dataset as a benchmark, to compare our newly proposed architectures with standard Prefix Tuning [18] and Control Prefixes [6] using a standardized test suite. This makes our scores more directly comparable with other results coming from the research community. We now discuss in detail some critical points that emerged from the experiments of Section 6.3.

Layer Dependency When we introduced layer dependency in Chapter 4, we said that the main idea was to introduce a prior, telling the model that the keys and values of the prefix at each layer of the transformer should depend on those before it. Our results seem to imply that such a prior has the desired effect only up to a point: indeed, models using our layer dependency typically have higher BLEU scores from the very first epochs, but the performances soon saturate. This is especially true for smaller, less expressive models. Overall, we can state that layer dependency seems to have a positive impact on performance when training with a limited computational budget, but that, when the number of epochs we train for increases, the benefits start to lessen. We also noticed a small boost in performances in Control Prefixes with respect to Prefix Tuning, leading us to believe that layer dependency is also introducing an explicit link between the static, task-specific prompt and the dynamic, example-dependent one. In practice, it is likely that using this new parametrization for Control Prefixes approaches does not only provide a prior but also encourages the sharing of information between the two prefix components.

Contrastive vs. Beam Search We also compared the results obtained by our prefix tuning models when varying the decoding strategy, as different techniques might produce vastly different verbalizations, thus having a big impact on performance. In particular, we focused on two methods, beam search, which currently achieves state-of-the-art results on data-to-text benchmarks [6, 18], and contrastive search, a novel technique that has, up until now, mainly been tested for open-ended text generation. Our results show that the latter is decidedly outperformed by beam search: we find that the BLEU score, in particular, is strongly affected by this change and drops by about 2 points in mean across all five models, while METEOR and TER are less impacted. The drop in performance is reasonable as Su et al. [46] introduce contrastive search for tasks like dialogue or poetry generation, where the aim is to produce a coherent continuation starting from a limited preceding context and without repetitions. In the data-to-text task, we are however solving a conditional generation problem, where repetitions might be welcome, if not required. It all depends on the structure of the input triplets. Thus, even keeping a low degeneralization penalty of 0.1, contrastive search runs the risk of forbidding multiple necessary occurrences of the same word. Su et al. [46] also point out that their decoding strategy works best on isotropic models, i.e., those models where all word vectors in the vocabulary are uniformly distributed with respect to direction. The idea is that, in anisotropic models, the cosine similarities between tokens all tend to be very high, because they are all located in a narrow portion of the whole space. This may cause the generative process to degenerate, producing repetitive tokens at multiple steps. Isotropy has however not been correctly evaluated yet for most of the generative pre-trained language models we use, as it has only recently been identified as a possible issue during the generation step [46].

Prefix Pooling In Section 4.3 we introduced Prefix Pooling, a new dynamic prompting technique that does not require any guidance signal. This could be extremely useful when no additional information is given together with the input, but we still know that the examples in our datasets might have different complexities. For instance, we pointed out how, in our data-to-text task, it is typically much harder to correctly verbalize a longer triplet set. Table 7.1 shows a more direct comparison between the performances of two models, that is, `t5-base` and `bart-large`, on our three main settings - Prefix Tuning, Control Prefixes, and Prefix Pooling. As we can see, Prefix Pooling can gener-

Table 7.1: Comparison between Prefix Tuning, Control Prefixes and Prefix Pooling for two of our models, `t5-base` and `bart-large`.

	BLEU ↑			**METEOR ↑**			**TER ↓**		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
				`t5-base`					
Prefix Tuning	64.14	46.88	56.46	**0.46**	**0.39**	**0.43**	**0.33**	0.48	0.40
Control Prefixes	64.01	45.97	55.93	**0.46**	**0.39**	0.42	0.34	0.48	0.41
Prefix Pooling	**64.50**	**47.50**	**56.86**	**0.46**	**0.39**	**0.43**	**0.33**	**0.46**	**0.39**
				`bart-large`					
Prefix Tuning	**63.88**	48.50	56.96	**0.45**	**0.40**	**0.43**	0.34	0.45	0.39
Control Prefixes	63.51	45.00	55.25	**0.45**	0.38	0.42	0.34	0.47	0.40
Prefix Pooling	63.14	**50.45**	**57.44**	**0.45**	**0.40**	0.42	**0.33**	**0.43**	**0.38**

ally achieve higher performances than Control Prefixes, while not needing any additional information to be associated with the input triplet sets. In particular, it is always the technique resulting in lower TER scores, indicating that a small number of edits are needed to recover a reference verbalization from the generation. Moreover, Prefix Pooling also outperforms or obtains results on par with Prefix Tuning: in particular, it seems to generalize well to new categories, obtaining better BLEU and TER scores for the unseen portion of the test set. We should also note that Prefix Pooling can be easily used in a purely dynamic way, without any task-specific prefix, or it can be combined with Prefix Tuning and/or Control Prefixes. This makes the whole architecture very flexible and allows for controlling the generation in a purely dynamic setting.

7.2 USMLE-Symp: Analysis of Results

The main objective of this work is to provide discoursive, symptom-based justifications for a medical diagnosis given both the clinical case and some information about the characterizing symptoms of the disease we are examining. We propose to frame this problem as a data-to-text task and solve it using prompting techniques: in particular, we experiment with current-state-of-the-art techniques like Control Prefixes [6] and Prefix Tuning [18], as well as with our newly proposed Prefix Pooling approach. We now highlight some crucial points that emerged from the experiments we detailed in Section 6.4.

General-purpose vs. Domain specific We hypothesized that, in tasks like data-to-text, where our main objective is to re-formulate in a natural, fluent, and coherent way information that is already present in the input, domain knowledge is not as important as linguistic knowledge. Indeed, our results in the Prefix Tuning setting show that general-purpose PLMs perform almost as well as those pre-trained on PubMed's abstracts or full-text articles. We however note that our medical models have been designed explicitly for text generation, so that they all maintain a good understanding of the English language and its grammar. Because of this, even if the difference in performance was limited, domain-specific models still outperformed their general-purpose counterparts. In particular, medically-oriented models obtain higher BLEU scores on the validation set and typically find it easier to avoid hallucinations and symptoms repetitions. They also are better in general at capturing the correct polarity of the generated text.

Control Prefixes After showing that domain-specific models typically provide a small boost in performance with respect to their general-purpose counterparts, we experimented with different guidance signal possibilities for the Control Prefixes approach. Indeed, our USMLE-Symp dataset, as described in Section 5.2, provides both polarity and category information associated with each example. Manually analyzing the produced verbalizations showed that even the clinical models sometimes struggle to come up with the correct polarity, especially for what concerns longer and more complex triplet sets. We thus hypothesized that introducing a related control prefix might produce improvements on this aspect. Notice that we can assume to always have this additional information available for every new triplet set. Indeed, positive or negative polarity is actually a desired output's characteristic more than an additional descriptive attribute for the input. Moreover, it is also true that, in the symptom-based diagnosis justification system that is being developed as part of the ANTIDOTE project [1], once the information is passed to the explanation generation step we already know which polarity the output justification should have. We thus first tried using polarity as a guidance signal and obtained improvements of at least 0.20 validation BLEU points for all our models. What is more important, qualitative analysis of the results showed a significant improvement in polarity alignment for both `bioGPT` and `biobart-large`.

Next, we tried to use the HPO category associated with the main disease/symptom in the input triplet set as a guidance signal, following Clive et al. [6] that report a noticeable improvement in performances on WebNLG when using this additional information. We however should note that our dataset has a significant difference from WebNLG, that is, the predicates we use are very standardized and mostly do not change with the triplets' category. Because of this, the second form guidance signal does not produce any significant improvement in terms of validation BLEU scores, nor in terms of quality of the generation. Instead, it seems to confuse the models, probably because it is indicating that two instances with different HPO categories should be treated differently somewhat, but both the triplet sets and the relative verbalizations have a very similar structure.

Finally, despite not having found any benefit from the use of the HPO category as a guidance signal, we experimented with combining category and polarity. We mainly wanted to verify whether an interaction between the two types of control prefixes could be beneficial to our generations. Indeed, we noticed an improvement in validation BLEU, especially for what concerns `bioGPT`, which obtains its highest scores with this configuration. However, a qualitative analysis of the results shows that they are quite similar to those produced by Control Prefixes with category-only guidance signal (i.e., they are missing a lot of crucial information). Thus, once again, the addition of category information seems to hurt performances instead of promoting coherence and fluency. Also notice that we have another example of the unreliability of automatic scores, as the validation BLEU for `bioGPT` is approximately 0.20 points higher than it was for polarity-only Control Prefixes, but the semantics of the produced generations do not reflect this improvement.

Comparison between Prompting techniques We now compare our three prompting techniques, that is, Prefix Tuning, Control Prefixes, and Prefix Pooling. In particular Table 7.2 shows the results obtained by both `bioGPT` and `biobart-large` in terms of test BLEU, METEOR, and TER for each of the approaches. The Control Prefixes results refer to the polarity-only setting, which achieves the overall best results on the validation set among the possibilities we tried. We note that, for `bioGPT`, Prefix Pool-

Table 7.2: Comparison of test results on USMLE-Symp dataset.

	BLEU ↑	**METEOR ↑**	**TER ↓**
`bioGPT`			
Prefix Tuning	96.26	0.982	3.46
Control Prefixes	**97.63**	**0.989**	**2.39**
Prefix Pooling	96.92	0.985	2.98
`biobart-large`			
Prefix Tuning	**96.78**	**0.986**	**2.78**
Control Prefixes	96.08	0.981	3.45
Prefix Pooling	96.37	0.980	3.59

ing outperforms Prefix Tuning in terms of automatic metrics while, from a qualitative point of view, the generated verbalizations are pretty similar. Control Prefixes with polarity information instead achieves the best results both qualitatively and quantitatively. For `biobart-large`, both dynamic prompting approaches are outperformed by standard Prefix Tuning. Notice that this might also be partially due to overfitting, as our dataset is quite small, the model `biobart-large` is bigger than `bioGPT` and the dynamic prefixes add degrees of complexity to our architecture.

Dataset Limitations We now acknowledge some limitations of the USMLE-Symp dataset due to its construction process. First of all, the verbalizations are generated automatically starting from the triplet components and using the templates shown in Section 5.2. This means that there is not a lot of variation and the dataset appears quite easy to solve. Crowdsourced annotations like those in WebNLG obviously produce a more challenging dataset; however, the process can be long and tedious. Moreover, due to the particular domain we are operating in, annotators with specific medical knowledge would have to be found. We also note that the category for a given triplet set was taken from HPO whenever a match between its disease/symptoms was found, but when no match was present we used the closest neighbor in the embedding space among the categories already found. This process was done automatically using a clinical version of BERT to compute the embeddings and the assignments have not been medically verified: eventual errors in this step might also contribute to explaining why category information does not improve performances when used as a guidance signal in Control Prefixes. Finally, we note that some of the more complex input triplet sets have been associated with an explanation that does not make use of all their components. Some examples of this situation are shown in our qualitative analysis in Appendix B.2. Often, the discarded information is related to patient symptomatology that is not characteristic of the main disease we are describing. The poor quality of the reference verbalization for these kinds of examples, together with the fact that they constitute a small portion of the training data, contributes to lower our models' performances.

Chapter 8

Conclusions

In this thesis, we experimented with prompting techniques for natural language generation in the medical domain. In particular, our aim was to generate fluent and discoursive justifications for a diagnosis given all the information about the patient that was present in the case description and all the symptoms linked with their disease. We framed the problem as a data-to-text task and solved it using lightweight prompting techniques like Prefix Tuning [18] and Control Prefixes [6]. We also proposed layer dependency, a straightforward modification to the way in which the prefix is parametrized that introduces an explicit dependency between the prompts used at different levels of the PLM. We proved through our experiments that layer dependency is able to produce a boost in performance when training with a limited computational budget, especially for smaller, less expressive models. For instance, it improves the validation BLEU score for `t5-base` by approximately 0.60 points when training for only four epochs. Our second contribution is a new dynamic prompting technique called Prefix Pooling, which is able to achieve performances on par with both Prefix Tuning and Control Prefixes, and even outperforms them on occasion, while not requiring any form of additional information.

Our techniques obtain impressive results on the medical USMLE-Symp dataset: the best model is `bioGPT` with control prefixes based on the polarity information, which achieves 97.63 BLEU points on the test set. Despite this, we should note that the dataset we use for both training and testing is based on template explanations whose quality may be lacking, especially when the number of triplets in the input set grows. The fact that all our gold standards are pattern-based plays an important role in partially explaining the excellent results our models are obtaining. Indeed, the sophisticated language models we are employing are surely good enough to learn how to mimic a set of templates. Thus, to verify the robustness of such a system before a real prototype can be deployed, it is important to gather a more complex and interesting dataset. Several possibilities are available, ranging from crowdsourcing to expert annotations, but we should point out another interesting possibility: since our USMLE-Symp data contains case descriptions collected from professional medical board exams, online studying and revising tools like Quizlet[1] offer a huge amount of material already solved by medical students, with the relative explanations. This could be an interesting approach to collect more discoursive and fluent ground truth justifications, even if it does not give any quality guarantee.

It would be valuable to also perform additional experiments with our proposed Prefix Pooling strategy, in order to assess its performances on more complex generative tasks like abstractive summarization or machine translation. Finally, we still did not extensively test the flexibility of this architecture: the prompts are now chosen from the pool based on their key's similarity with the input embeddings. However, these embeddings could also be pre-processed somehow: we could for example pass them through some network layers to extract more meaningful and high-level information. Moreover, we only tested Prefix Pooling in conjunction with a static, task-specific prompt, while we know that the modular way in which it is designed makes it possible to use it alone, in a fully-dynamic setting, or combine it with Control Prefixes to achieve controllable generation.

[1]https://quizlet.com/

Bibliography

[1] ANTIDOTE (ArgumeNtaTIon-Driven explainable artificial intelligence fOr digiTal mEdicine). en, December 2020. URL: https://www.chistera.eu/projects/antidote (visited on 11/30/2022).

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: a nucleus for a web of open data. In K. Aberer, K.-S. Choi, N. Noy, et al., editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-76298-0.

[3] S. Banerjee and A. Lavie. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics, June 2005. URL: https://aclanthology.org/W05-0909.

[4] T. B. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners, 2020. DOI: 10.48550/ARXIV.2005.14165. URL: https://arxiv.org/abs/2005.14165.

[5] S. Chen, Y. Hou, Y. Cui, W. Che, T. Liu, and X. Yu. Recall and learn: fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics, November 2020. DOI: 10.18653/v1/2020.emnlp-main.634. URL: https://aclanthology.org/2020.emnlp-main.634.

[6] J. Clive, K. Cao, and M. Rei. Control prefixes for parameter-efficient text generation. In *Proceedings of the 2nd Workshop on Natural Language Generation,*

Evaluation, and Metrics (GEM), pages 363–382, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics, December 2022. URL: https://aclanthology.org/2022.gem-1.31.

[7] K. yras, A. Rago, E. Albini, P. Baroni, and F. Toni. Argumentative xai: a survey, 2021. DOI: 10.48550/ARXIV.2105.11266. URL: https://arxiv.org/abs/2105.11266.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. DOI: 10.1109/CVPR.2009.5206848.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June 2019. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[10] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics, July 2017. DOI: 10.18653/v1/P17-1017. URL: https://www.aclweb.org/anthology/P17-1017.pdf.

[11] X. Han, Z. Zhang, N. Ding, et al. Pre-trained models: past, present and future, 2021. DOI: 10.48550/ARXIV.2106.07139. URL: https://arxiv.org/abs/2106.07139.

[12] T. He, J. Liu, K. Cho, M. Ott, B. Liu, J. Glass, and F. Peng. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association*

for Computational Linguistics: Main Volume, pages 1121–1133, Online. Association for Computational Linguistics, April 2021. DOI: 10.18653/v1/2021.eacl-main.95. URL: https://aclanthology.org/2021.eacl-main.95.

[13] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, September 2019. URL: https://proceedings.mlr.press/v97/houlsby19a.html.

[14] D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy. Association for Computational Linguistics, July 2019. DOI: 10.18653/v1/P19-1365. URL: https://aclanthology.org/P19-1365.

[15] S. Köhler, M. Gargano, N. Matentzoglu, et al. The human phenotype ontology in 2021. en. *Nucleic Acids Res.*, 49(D1):D1207–D1217, January 2021.

[16] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics, July 2020. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

[17] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen. Pretrained language models for text generation: a survey, 2022. DOI: 10.48550/ARXIV.2201.05273. URL: https://arxiv.org/abs/2201.05273.

[18] X. L. Li and P. Liang. Prefix-tuning: optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics, August 2021. DOI: 10.18653/v1/2021.acl-long.353. URL: https://aclanthology.org/2021.acl-long.353.

[19] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing, 2021. DOI: 10.48550/ARXIV.2107.13586. URL: https://arxiv.org/abs/2107.13586.

[20] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too, 2021. DOI: 10.48550/ARXIV.2103.10385. URL: https://arxiv.org/abs/2103.10385.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: a robustly optimized bert pretraining approach, 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

[22] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu. BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, September 2022. ISSN: 1477-4054. DOI: 10.1093/bib/bbac409. eprint: https://academic.oup.com/bib/advance-article-pdf/doi/10.1093/bib/bbac409/45989562/bbac409.pdf. URL: https://doi.org/10.1093/bib/bbac409.

[23] S. Marro, B. Molinet, E. Cabrio, and S. Villata. Natural Language Explanatory Arguments for Correct and Incorrect Diagnoses of Clinical Cases. In *ICAART 2023 - 15th International Conference on Agents and Artificial Intelligence*, volume 1, pages 438–449, Lisbon (Portugal), France, February 2023. URL: https://hal.science/hal-04002207.

[24] T. Mayer, S. Marro, E. Cabrio, and S. Villata. Enhancing evidence-based medicine with natural language argumentative analysis of clinical trials. *Artificial Intelligence in Medicine*, 118:102098, 2021. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2021.102098. URL: https://www.sciencedirect.com/science/article/pii/S0933365721000919.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Lake Tahoe, Nevada. Curran Associates Inc., 2013.

[26] B. Molinet, S. Marro, E. Cabrio, S. Villata, and T. Mayer. Acta 2.0: a modular architecture for multi-layer argumentative analysis of clinical trials. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5940–5943. International Joint Conferences on Artificial Intelligence Organization, July 2022. DOI: 10.24963/ijcai.2022/859. URL: https://doi.org/10.24963/ijcai.2022/859. Demo Track.

[27] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics, August 2016. DOI: 10.18653/v1/K16-1028. URL: https://aclanthology.org/K16-1028.

[28] S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics, October 2018. DOI: 10.18653/v1/D18-1206. URL: https://aclanthology.org/D18-1206.

[29] K. Oberauer. Working Memory and Attention A Conceptual Analysis and Review. *Journal of Cognition*, 2(1):36. ISSN: 2514-4820. DOI: 10.5334/joc.58. URL:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6688548/ (visited on 12/14/2022).

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics, July 2002. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[31] J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics, October 2014. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[32] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics, November 2019. DOI: 10.18653/v1/D19-1250. URL: https://aclanthology.org/D19-1250.

[33] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vuli, S. Ruder, K. Cho, and I. Gurevych. Adapterhub: a framework for adapting transformers, 2020. DOI: 10.48550/ARXIV.2007.07779. URL: https://arxiv.org/abs/2007.07779.

[34] L. N. Phan, J. T. Anibal, H. Tran, S. Chanana, E. Bahadroglu, A. Peltekian, and G. Altan-Bonnet. Scifive: a text-to-text transformer model for biomedical literature, 2021. arXiv: 2106.03598 [cs.CL].

[35] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. en. *Science China Technological Sciences*, 63(10):1872–1897, October 2020. ISSN: 1869-1900. DOI: 10.1007/s11431-020-

1647-3. URL: https://doi.org/10.1007/s11431-020-1647-3 (visited on 12/12/2022).

[36] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving Language Understanding by Generative Pre-Training. en.

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[38] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL: http://jmlr.org/papers/v21/20-074.html.

[39] W. S. Richardson, M. C. Wilson, J. Nishikawa, and R. S. Hayward. The well-built clinical question: a key to evidence-based decisions. eng. *ACP journal club*, 123(3):A12–13, 1995. ISSN: 1056-8751.

[40] D. Rubinstein, E. Levi, R. Schwartz, and A. Rappoport. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 726–730, Beijing, China. Association for Computational Linguistics, July 2015. DOI: 10.3115/v1/P15-2119. URL: https://aclanthology.org/P15-2119.

[41] A. B. Sai, A. K. Mohankumar, and M. M. Khapra. A survey of evaluation metrics used for nlg systems. *ACM Comput. Surv.*, 55(2), January 2022. ISSN: 0360-0300. DOI: 10.1145/3485766. URL: https://doi.org/10.1145/3485766.

[42] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics, August 2016. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

[43] N. Shazeer and M. Stern. Adafactor: adaptive learning rates with sublinear memory cost, 2018. DOI: 10.48550/ARXIV.1804.04235. URL: https://arxiv.org/abs/1804.04235.

[44] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, 2020.

[45] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas, August 2006. URL: https://aclanthology.org/2006.amta-papers.25.

[46] Y. Su, T. Lan, Y. Wang, D. Yogatama, L. Kong, and N. Collier. A contrastive framework for neural text generation, 2022. DOI: 10.48550/ARXIV.2202.06417. URL: https://arxiv.org/abs/2202.06417.

[47] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[48] A. Vassiliades, N. Bassiliades, and T. Patkos. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*, 36:e5, 2021. DOI: 10.1017/S0269888921000011.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[50] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: a multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics, November 2018. DOI: 10.18653/v1/W18-5446. URL: https://aclanthology.org/W18-5446.

[51] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister. Learning to prompt for continual learning, 2021. DOI: 10.48550/ARXIV.2112.08654. URL: https://arxiv.org/abs/2112.08654.

[52] T. Wolf, L. Debut, V. Sanh, et al. Transformers: state-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics, October 2020. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[53] H. Yuan, Z. Yuan, R. Gan, J. Zhang, Y. Xie, and S. Yu. BioBART: pretraining and evaluation of a biomedical generative language model. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 97–109, Dublin, Ireland. Association for Computational Linguistics, May 2022. DOI: 10.18653/v1/2022.bionlp-1.9. URL: https://aclanthology.org/2022.bionlp-1.9.

Appendices

Appendix A

Hyperparameters

All the presented architectures are built on top of HuggingFace's transformers library [52]. For Prefix Tuning, we adapted the original implementation [18] to work with version 4.26.0 of the library and extended the GPT-2 implementation to work with BioGPT [22]. Similarly, our Control Prefixes architecture follows the implementation of Clive et al. [6]. The following section show the hyperparameters we use to train our models for the experiments we discussed in Chapter 6. Also notice that, whenever the generation is performed with beam search, we use the parameters shown in table A.1, while for contrastive search we use those displayed in Table A.2.

Table A.1: Beam search decoding parameters

Parameter	Value
Max length	300
Early stop	True
Num beams	5
Do sample	False

Table A.2: Contrastive search decoding parameters

Parameter	Value
Max length	300
Early stop	True
Top k	5
Penalty α	0.1

A.1 Prefix Tuning

Table A.3 shows in detail the hyperparameters used to train our Prefix Tuning models. Note that the length of the task-specific prompt and the choice of hidden dimension for the prefix are taken from the literature [18, 6]. We train for 7 epochs for the bigger and more complex WebNLG dataset and train for 4 epochs - using layer dependency to boost performances - when working with the USMLE-Symp dataset.

Table A.3: Detailed report of the hyperparameters for the prefix tuning models in this work. L-rate stands for learning rate.

	Model				
	`t5-base/ SciFive-base`	`t5-large`	`gpt2-medium/ bioGPT/`	`gpt2-large`	`bart-large biobart-large`
L-rate	7e-5	7e-5	1e-4	1e-4	1e-4
Optimizer	Adafactor	Adafactor	AdamW	AdamW	AdamW
Warm-up	2000	2000	0	0	0
Epochs	[7,4]	7	[7,4]	7	[7,4]
Batch size	5	5	5	5	5
Prefix len	48	48	5	5	5
Hidden dim	800	800	512	512	512
T5-preamble	✓	✓	-	-	-
Dropout	0.0	0.0	0.0	0.0	0.0
Weight decay	0.0	0.0	0.0	0.0	0.0

A.2 Control Prefixes

Table A.4 shows in detail the hyperparameters we used to train all control prefixes models. Note that the length of the dynamic prompt is taken from the literature [6] and that we never rely just on the control prefixes but always pair them with the task-specific prefix. We also do not use any additional token to indicate the start of every triplet component. We again train for 7 epochs on the bigger WebNLG dataset and for

Table A.4: Detailed report of the hyperparameters for the control prefixes models in this work. L-rate stands for learning rate.

	Model				
	`t5-base/ SciFive-base`	`t5-large`	`gpt2-medium/ bioGPT/`	`gpt2-large`	`bart-large biobart-large`
L-rate	7e-5	7e-5	1e-4	1e-4	1e-4
Optimizer	Adafactor	Adafactor	AdamW	AdamW	AdamW
Warm-up	2000	2000	0	0	0
Epochs	[7,4]	7	[7,4]	7	[7,4]
Batch size	5	5	5	5	5
Prefix len	[48, 46]	48	[5,3]	5	[5,3]
Hidden dim	800	800	512	512	512
T5-preamble	✓	✓	-	-	-
Dropout	0.0	0.0	0.0	0.0	0.0
Weight decay	0.0	0.0	0.0	0.0	0.0
C-prefix Len	2	2	2	2	2

4 epochs with layer dependency on the USMLE-Symp dataset. In Table A.4, when a list of values is reported for one hyperparameter, they refer to those used for the general purpose and for the medical portion of this work, respectively.

A.3 Prefix Pooling

Table A.5 shows a detailed report of the hyperparameters used to train all our prefix pooling models. The pool size and the length of the dynamic prompt, as well as the number of dynamic components to use, are derived through preliminary experimentations literature [6]. Finally, notice that we never rely on a learned key vector for every prompt in the pool as preliminary experiments showed it to cause overfitting. As previously pointed out, we train for a maximum of 7 epochs when using WebNLG and for a maximum of 4 epochs on USMLE-Symp, as it is a much smaller dataset. For the same reason we also slightly decreased the length of the prefix. Then, in Table A.5, when a list of values is present for a certain hyperparameter, the first one represents the configuration used for WebNLG while the second represents the value choice for USMLE-Symp.

Table A.5: Detailed report of the hyperparameters for the prefix pooling models in this work. L-rate stands for learning rate.

	Model				
	`t5-base/ SciFive-base`	`t5-large`	`gpt2-medium/ bioGPT/`	`gpt2-large`	`bart-large biobart-large`
L-rate	7e-5	7e-5	1e-4	1e-4	1e-4
Optimizer	Adafactor	Adafactor	AdamW	AdamW	AdamW
Warm-up	2000	2000	0	0	0
Epochs	[7,4]	7	[7,4]	7	[7,4]
Batch size	5	5	5	5	5
Prefix len	[48,46]	48	[5,3]	5	[5,3]
Hidden dim	800	800	512	512	512
T5-preamble	✓	✓	-	-	-
Dropout	0.0	0.0	0.0	0.0	0.0
Weight decay	0.0	0.0	0.0	0.0	0.0
Pool size	10	10	10	10	10
Top-k	3	3	3	3	3
P-prefix len	2	2	2	2	2

Appendix B

Qualitative Analysis

Here we present some examples of the output produced by our models, with a particular focus on highlighting benefits and drawbacks of our architectural changes. It is important to visualize the generated verbalizations because it is notoriously problematic to comprehensively evaluate the performances of a generative language model relying simply on automatic metrics like BLEU and TER. Indeed, in a data-to-text task, we are concerned with obtaining fluent text that covers all the information provided by the source and does not hallucinate knowledge. Unfortunately, automatic metrics are often based on n-gram overlap or on the edit distance between reference and generation, thus not considering the semantics. This implies that we might obtain a quite good BLEU score while still having a generation that mixes up information or completely negates the original triplets' content.

B.1 WebNLG Human Evaluation

We now report some examples coming from the test dataset in WebNLG. In particular, Tables B.1 and B.2 show the verbalizations generated by each of our five models (i.e., `gpt2-medium`, `gpt2-large`, `t5-base`, `t5-large` and `bart-large`) in the Prefix Tuning setting for two triplets with seen categories *Astronaut* and *Building*. Note that all our T5 implementations here use the textual preamble "translate Graph to English: ", as specified in Section 6.3. Moreover, for each model, we make a point of signaling with a checkmark the use of layer dependency or not, in order to check whether this modification has a tangible impact on performance.

Table B.1: Generation example from the WebNLG seen category *Astronaut* in prefix tuning setting. The checkmark (✓) indicates the use of layer dependency.

Triplet	(William_Anders, dateOfRetirement, "1969-09-01"), (William_Anders, was a crew member of, Apollo_8), (William_Anders, nationality, United_States), (William_Anders, birthPlace, British_Hong_Kong), (Apollo_8, backup pilot, Buzz_Aldrin), (Apollo_8, crewMembers, Frank_Borman), (Apollo_8, operator, NASA)
Reference	William Anders was born in British Hong Kong and is a U.S Citizen. William was a member of the Apollo 8 crew (along with Frank Borman) which was operated by NASA's backup pilot Buzz Aldrin. William retired on September 1st in 1969.
gpt2-medium ✓	American William Anders was born in British Hong Kong. He was a crew member of Apollo 8 operated by NASA. The backup pilot was Buzz Aldrin and Frank Borman was a crew member. William Anders retired on September 1, 1969.
gpt2-medium	American William Anders was born in British Hong Kong. He was a crew member of Apollo 8 which was operated by NASA. Buzz Aldrin was a backup pilot and Frank Borman was a crew member. William Anders retired on September 1, 1969.
gpt2-large ✓	American William Anders was born in British Hong Kong. He was a member of the Apollo 8 crew along with Frank Borman and backup pilot Buzz Aldrin. William Anders retired on the 1st of September 1969.
gpt2-large	American William Anders was born in British Hong Kong and is a citizen of the United States. He served as a crew member on the NASA operated Apollo 8 along with backup pilot Buzz Aldrin and Frank Borman. He retired on September 1, 1969.
t5-base ✓	American William Anders was born in British Hong Kong. He was a member of NASA's Apollo 8 crew, along with Frank Borman and Buzz Aldrin. He retired on September 1st 1969.
t5-base	William Anders was born in British Hong Kong. He was part of NASA's Apollo 8 crew along with Frank Borman and Buzz Aldrin. He retired on September 1st, 1969.
t5-large ✓	American William Anders was born in British Hong Kong. He was a crew member of Apollo 8 operated by NASA. Buzz Aldrin was a backup pilot on Apollo 8 and Frank Borman was a crew member. Anders retired on September 1st, 1969.
t5-large	American William Anders was born in British Hong Kong. He was a crew member of Apollo 8 operated by NASA. Buzz Aldrin was a backup pilot and Frank Borman was a crew member. He retired on September 1st, 1969.
bart-large ✓	William Anders was born in British Hong Kong and was a member of NASA's Apollo 8 crew along with Buzz Aldrin as backup pilot and Frank Borman as a crew member. William Anders retired on 1969-09-01.
bart-large	William Anders, born in British Hong Kong, was a member of the NASA operated Apollo 8 crew along with Frank Borman and Buzz Aldrin. William Anders retired on September 1st, 1969.

We can see that there are significant changes in generation quality between models: for instance, `gpt2-large` in Table B.2 is missing crucial information about the country where the hotel is located, no matter whether we use layer dependency or not. Similarly, `bart-large` in Table B.1 is missing details about the nationality of William Anders. We also note that, in general, using layer dependency does not give rise to huge differences in the verbalizations. However, it can sometimes produce improvements for smaller models like `t5-base` and `gpt2-medium`. Indeed, in Table B.1, `t5-base` with standard Prefix Tuning omits the astronaut's nationality, while our different parametrization seems to recover this detail. In the same table, `gpt2-medium` with layer dependency avoids the repetition of Sanders' nationality.

Table B.2: Generation example from the WebNLG seen category *Building* in prefix tuning setting. The checkmark (✓) indicates the use of layer dependency.

Triplet	(AC_Hotel_Bella_Sky_Copenhagen, location, Denmark), (AC_Hotel_Bella_Sky_Copenhagen, tenant, Marriott_International), (AC_Hotel_Bella_Sky_Copenhagen, floorCount, 23)
Reference	AC Hotel Bella Sky Copenhagen has 23 floors and is located in Denmark. The tenant is Marriott International
gpt2-medium ✓	Marriott International is the tenant of AC Hotel Bella Sky Copenhagen which has 23 floors and is located in Denmark.
gpt2-medium	Marriott International is a tenant of AC Hotel Bella Sky Copenhagen which is located in Denmark and has 23 floors.
gpt2-large ✓	Marriott International is the tenant of AC Hotel Bella Sky Copenhagen, which has 23 floors.
gpt2-large	Marriott International is the tenant of AC Hotel Bella Sky Copenhagen which has 23 floors.
t5-base ✓	Marriott International is a tenant of AC Hotel Bella Sky Copenhagen which has 23 floors and is located in Denmark.
t5-base	Marriott International is a tenant of AC Hotel Bella Sky Copenhagen which is located in Denmark and has 23 floors.
t5-large ✓	Marriott International is the tenant of AC Hotel Bella Sky Copenhagen, Denmark which has 23 floors.
t5-large	Marriott International is the tenant of AC Hotel Bella Sky Copenhagen which has 23 floors and is located in Denmark.
bart-large ✓	Marriott International is a tenant of the AC Hotel Bella Sky Copenhagen which has 23 floors and is located in Denmark
bart-large	Marriott International is a tenant of AC Hotel Bella Sky Copenhagen which is located in Denmark and has 23 floors.

Next, Tables B.3–B.7 show a comparison of the verbalizations generated by models `t5-large` and `bart-large` for input triplet sets coming from each of the five unseen test categories. In Table B.6 we can quite clearly notice how both models are unfamiliar with the astronomy concept of an "epoch date": they thus interpret it as a generic calendar day. It is worth mentioning that we can see the impact of the training categories: indeed, both our models treat the asteroid as something that can be established/created (like a company) or as a book that can be published.

Table B.3: Generation example from the WebNLG unseen category *MeansOfTransportation*.

Triplet	(Aston_Martin_V8, relatedMeanOfTransportation, Aston_Martin_DBS), (Aston_Martin_V8, engine, 5.3 (litres)), (Aston_Martin_V8, successor, Aston_Martin_Virage), (Aston_Martin_Virage, manufacturer, Aston_Martin)
Reference	The Aston Martin V8, manufactured by Aston Martin, has a 5.3 litre engine and was succeeded by the Aston Martin Virage. The Aston Martin V8 and Aston Martin DBS are a related means of transport.
	`t5-large`
Prefix Tuning	Aston Martin DBS is a successor to the Aston Martin V8 which has a 5.3 litre engine and was made by Aston Martin. Aston Martin Virage is a successor to the Aston Martin V8.
Control Prefixes	Aston Martin DBS is the manufacturer of the Aston Martin V8, which is the successor to the Aston Martin Virage. The V8 has a 5.3 litre engine and is related to the Aston Martin V8, which is also a means of transport.
Prefix Pooling	The Aston Martin DBS is a successor to the Aston Martin V8 which has a 5.3 litre engine. The Aston Martin Virage was also made by Aston Martin.
	`bart-large`
Prefix Tuning	Aston Martin DBS is the manufacturer of Aston Martin V8, which has a 5.3 litre engine. The V8 is a successor of the Aston Martin Virage.
Control Prefixes	The Aston Martin DBS is the manufacturer of the Aston Martin V8, which has an engine of 5.3 litres.
Prefix Pooling	Aston Martin DBS is the manufacturer of Aston Martin V8 which has a 5.3 litre engine.

We should also note that the example in Table B.3 is longer than the others and thus harder to correctly verbalize: almost all model variations apart from `t5-large` Prefix Tuning and the two Prefix Pooling models consider the Aston Martin DBS as the manufacturer of the Aston Martin V8, instead of correctly considering it as a *related means of transportation*. Finally, we also point out how, in Table B.5, only `bart-large` with Prefix Pooling correctly interprets the *successor* relation between Alberto Tessaire and Isaac Rojas.

Table B.4: Generation example from the WebNLG unseen category *Artist*.

Triplet	(Asunción, isPartOf, Gran_Asunción), (Agustín_Barboza, deathPlace, Asunción), (Paraguay, leaderName, Juan_Afara), (Agustín_Barboza, deathPlace, Paraguay)
Reference	Agustin Barboza died in Asuncion, part of Gran Asunción, in Paraguay, the country led by Juan Afara.
	`t5-large`
Prefix Tuning	Agustn Barboza died in Asunción, Gran Asunción, Paraguay, where the leader is Juan Afara.
Control Prefixes	Agustn Barboza died in Paraguay where the leader is Juan Afara. Asunción is part of Gran Asunción.
Prefix Pooling	Agustn Barboza died in Asunción, which is part of Gran Asunción. The leader of Paraguay is Juan Afara.
	`bart-large`
Prefix Tuning	Agustín Barboza died in Asunción, part of Gran Asuncion. The leader of Paraguay is Juan Afara.
Control Prefixes	Agustín Barboza is the death place of Juan Afara who is the leader of Paraguay. Asunción is part of Gran Asuncion.
Prefix Pooling	Agustín Barboza died in Asunción, which is part of Gran Asuncion, in Paraguay, where Juan Afara is the leader.

Table B.5: Generation example from the WebNLG unseen category *Politician*. Comparison of various approaches for `t5-large` and `bart-large`.

Triplet	(Alberto_Teisaire, office (workedAt, workedAs), “Provisional President of the Argentine Senate“), (Alberto_Teisaire, successor, Isaac_Rojas)
Reference	Alberto Teisaire worked as the Provisional President of the Argentine Senate and was succeded by Isaac Rojas.
	`t5-large`
Prefix Tuning	Alberto Teisaire succeeded Isaac Rosas as the Provisional President of the Argentine Senate.
Control Prefixes	Alberto Teisaire served as the Provisional President of the Argentine Senate and succeeded Isaac Rojas.
Prefix Pooling	Alberto Teisaire succeeded Isaac Rojas as the Provisional President of the Argentine Senate.
	`bart-large`
Prefix Tuning	Alberto Teisaire worked at the office of Provisional President of the Argentine Senate and is the successor of Isaac Rojas.
Control Prefixes	Alberto Teisaire, who is the successor to Isaac Rojas, is the Provisional President of the Argentine Senate.
Prefix Pooling	Alberto Teisaire is the Provisional President of the Argentine Senate and his successor is Isaac Rojas.

Table B.6: Generation example from the WebNLG unseen category *CelestialBody*.

Triplet	(10_Hygiea, formerName, “A900 GA”), (10_Hygiea, epoch, 2015-06-27)
Reference	The asteroid 10 Hygiea, once called A900 GA, has the epoch date of 27th June 2015.
	`t5-large`
Prefix Tuning	10 Hygiea (previously known as A900 GA) was created on 2015-06-27.
Control Prefixes	10 Hygiea, whose former name was A900 GA, was published on the 27th of June, 2015.
Prefix Pooling	10 Hygiea (A900 GA) was founded on 2015-06-27.
	`bart-large`
Prefix Tuning	A900 GA was the former name of 10 Hygiea which was discovered on the 27th of June, 2015.
Control Prefixes	A900 GA was the former name of 10 Hygiea which was established in 2015-06-27
Prefix Pooling	A900 GA was the former name of 10 Hygiea which was invented on the 27th of June 2015

Table B.7: Generation example from the WebNLG unseen category *Athlete*.

Triplet	(Aleksandre_Guruli, club, Olympique_Lyonnais), (Olympique_Lyonnais, ground, Parc_Olympique_Lyonnais), (Aleksandre_Guruli, club, US_Lesquin)
Reference	Aleksandre Guruli, whose club is US Lesquin, played for the Olympique Lyonnais club who have their home ground at Parc Olympique Lyonnais stadium.
	`t5-large`
Prefix Tuning	Aleksandre Guruli plays for Olympique Lyonnais, whose ground is the Parc Olympique Lyonnais. He is also part of the US Lesquin club.
Control Prefixes	Aleksandre Guruli is attached to the club US Lesquin and plays for Olympique Lyonnais whose ground is Parc Olympique Lyonnais.
Prefix Pooling	Aleksandre Guruli has represented the club US Lesquin and plays for Olympique Lyonnais at their ground, Parc Olympique Lyonnais.
	`bart-large`
Prefix Tuning	Aleksandre Guruli played for US Lesquin and now plays for Olympique Lyonnais whose ground is Parc Olympique lyonnais.
Control Prefixes	Aleksandre Guruli is in the US Lesquin club and plays for Olympique Lyonnais whose ground is Parc Olympiis.
Prefix Pooling	Aleksandre Guruli plays at the Parc Olympique Lyonnais and has represented the club US Les Quins.

B.2 USMLE-Symp Human Evaluation

We now show some examples from the USMLE-Symp validation and test sets. In particular, Table B.8 shows the verbalizations generated by all our Prefix Tuning models apart from `t5-large` for a validation instance with positive polarity. Similarly, Table B.9 displays the produced descriptions for two easier validation triplet sets, composed of at most two elements each. Notice that `t5-large` is never included in our analysis because we previously showed (Section 6.4) how it obtains poorer automatic scores and produces lower quality generations. Examining the aforementioned tables, we see that, for the

Table B.8: Generation example from the USMLE-Symp validation set. The example is of category *Blood and blood-forming tissues* and has *positive* polarity

Triplet	(36-year-old woman, Suffers_from, Endometriosis), (36-year-old woman, Has_symptom, primary infertility), (36-year-old woman, Has_symptom, unable to conceive), (36-year-old woman, Has_symptom, dysmenorrhea), (Endometriosis, Has_as_symptom, dysmenorrhea)
Reference	The patient can be diagnosed with Endometriosis as they are showing the following symptoms, which are direct symptoms of Endometriosis: dysmenorrhea.
gpt2-medium	These symptoms indicate that the patient my suffer from Endometriosis: primary infertility, unable to conceive, dysmenorrhea and dysmenorrhea.
bioGPT	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive, dysmenorrhea and dysmenorrhea.
t5-base	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive, dysmenorrhea and dysmenorrhea.
SciFive-base	The 36-year-old woman has the following symptoms: Primary infertility, unable to conceive and dysmenorrhea.
bart-large	The 36-year-old woman has the following symptoms: dysmenorrhea and primary infertility.
biobart-large	TheThe 36-year-old woman has the following symptoms: endometriosis, primary infertility, unable to conceive and dysmenorrhea.

more complex triplet sets, the quality of the reference verbalization is quite poor. For instance, in Table B.8, the gold standard explains why the Endometriosis diagnosis is the correct one using *dysmenorrhea* as a main symptom because it was the only one for which a match was found in the HPO. However, no mention is made of the remaining patient symptoms which are part of the input triplet set. It is intuitive to understand how such misalignment between references and input may confuse our models.

Table B.9: Generation examples from the USMLE-Symp validation set. Both triplet sets are short, having at most two components.

Triplet	(patient, is_a, 11-year-old girl), (patient, Does_not_suffer_from, Exanthem subitum)
Reference	The patient is a 11-year-old girl and is not affected by Exanthem subitum.
gpt2-medium	The patient is a 11-year-old girl and is not affected by Exanthem subitum.
bioGPT	The patient is a 11-year-old girl and is not affected by Exanthey subitum.
t5-base	The patient, a 11-year-old girl, does not suffer from Exanthem subitum.
SciFive-base	The patient, a 11-year-old girl, does not suffer from Exanthem subitum.
bart-large	The patient, a 11-year-old girl, does not suffer from Exanthem subitum.
biobart-large	The patient, a 11-year-old girl, does not suffer from Exanthem subitum.
Triplet	(Huntington disease, Has_as_symptom, Decreased body mass index)
Reference	Decreased body mass index is a symptom of Huntington disease.
gpt2-medium	People with the disease Huntington may show Decreased body mass index as a symptom.
bioGPT	Decreased body mass index is a symptom of Huntington disease.
t5-base	Decreased body mass index is a symptom of Huntington disease.
SciFive-base	People with the disease Huntington may show Decreased body mass index as a symptom.
bart-large	People with the disease Huntington may show Decreased body mass index as a symptom.
biobart-large	Decreased body mass index is a symptom of Huntington disease.

Table B.10: Generation example from the USMLE-Symp validation set. The texts have been generated using Control Prefixes with *polarity* as the guidance signal.

Triplet	36-year-old woman, Suffers_from, Endometriosis), (36-year-old woman, Has_ symptom, primary infertility), ... , (Endometriosis, Has_as_symptom, dysmenorrhea)
Reference	The patient can be diagnosed with Endometriosis as they are showing the following symptoms, which are direct symptoms of Endometriosis: dysmenorrhea.
bioGPT	The 36-year-old woman has been diagnosed with Endometriosis and is affected by these symptoms: Primary infertility, Inable to conceive, Dysdysmenorrhea and Endometriosis.
SciFive-base	The 36-year-old woman suffers from Endometriosis: primary infertility, unable to conceive, dysmenorrhea and Endometriosis.
biobart-large	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive and dysmenorrhea.

Moreover, we should note that complex triplet sets like the *Endometriosis* one are very much a minority part of the training set. Because of this, our models' verbalizations tend to list the patient symptomatology, without reaching any conclusion about the diagnosis. Differently, for smaller examples such as those reported in Table B.9, all the generations achieve an almost perfect overlap with the gold standards. The fact that general-purpose models obtain results that are as good as those of the clinical versions, confirms our hypothesis that linguistic knowledge is much more important than domain-specific information when performing a data-to-text task, as all the necessary medical terminology is already included in the input.

Table B.11: Generation example from the USMLE-Symp validation set. The texts have been generated using both Control Prefixes and Prefix Pooling.

Triplet	36-year-old woman, Suffers_from, Endometriosis), (36-year-old woman, Has_ symptom, primary infertility), ... , (Endometriosis,Has_as_symptom, dysmenorrhea)
Reference	The patient can be diagnosed with Endometriosis as they are showing the following symptoms, which are direct symptoms of Endometriosis: dysmenorrhea.
CONTROL PREFIXES: CATEGORY-ONLY	
bioGPT	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive, dysmenorrhea and endometriosis.
SciFive-base	The 36-year-old woman suffers from Endometriosis, primary infertility, unable to conceive and dysmenorrhea.
biobart-large	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive and endometriosis.
CONTROL PREFIXES: ALL	
bioGPT	These symptoms indicate that the patient my suffer from Endometriosis: Primary infertility, unable to conceive, Endometriosis and dysmenorrhea.
SciFive-base	The 36-year-old woman has the following symptoms: Endometriosis, unable to conceive and dysmenorrhea.
biobart-large	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive, dysmenorrhea and Endometriosis.
PREFIX POOLING	
bioGPT	The 36-year-old woman has the following symptoms: Primary infertility, unable to conceive, dysmenorrhea and Endometriosis.
SciFive-base	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive, dysmenorrhea and endometriosis.
biobart-large	The 36-year-old woman has the following symptoms: primary infertility, unable to conceive and dysmenorrhea.

Table B.12: Generation example with negative polarity from USMLE-Symp test set.

Triplet	(Hemophilia A, Has_as_symptom, Reduced factor VIII activity), (Hemophilia A, Has_as_symptom, Joint hemorrhage), ..., (patient, is_a, 27-year-old man), (patient, Does_not_suffer_from, Hemophilia A)
Reference	The patient is not exhibiting Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding, hence the Hemophilia A diagnosis must be rejected.
	`bioGPT`
Prefix Tuning	The following list of symptoms is not found in the Hemophilia A patient: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.
Control Prefixes	The following list of symptoms is not found in the case description, hence the Hemophilia A diagnosis must be rejected: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.
Prefix Pooling	The patient, a 27-year-old man, has the following symptoms: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.
	`biobart-large`
Prefix Tuning	TheThe patient is a 27-year-old man and is affected by Hemophilia A: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.
Control Prefixes	The following list of symptoms is not found in the case description, hence the Hemophilia A diagnosis must be rejected: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.
Prefix Pooling	The patient is not exhibiting these symptoms: Reduced factor VIII activity, Joint hemorrhage, Bleeding with minor or no trauma, Thromboembolism and Oral cavity bleeding.

Table B.13: Second generation example with negative polarity from the USMLE-Symp test set.

Triplet	(patient, is_a, 32-year-old woman), (patient, Does_not_suffer_from, Acute leukemia)
Reference	The patient, a 32-year-old woman, does not suffer from Acute leukemia.
	`bioGPT`
Prefix Tuning	The patient is a 32-year-old woman and is not affected by Acute leukemia.
Control Prefixes	The patient is a 32-year-old woman and is not affected by Acute leukemia.
Prefix Pooling	The patient, a 32-year-old woman, does not suffer from Acute leukemia.
	`biobart-large`
Prefix Tuning	The patient, a 32-year-old woman, does not suffer from Acute leukemia.
Control Prefixes	The patient is a 32-year-old woman and is not affected by Acute leukemia.
Prefix Pooling	The patient, a 32-year-old woman, does not suffer from Acute leukemia.

We also point out that, in general, medical models generate more complete and fluent verbalizations, while avoiding unnecessary repetitions. For instance, in Table B.8 `SciFive-large` avoids the repetition of symptom *dysmenorrhea*. Tables B.12 to B.15 show four examples of varying degrees of complexity coming from the USMLE-Symp test dataset: we mainly focus on comparing the three prompting techniques one against the other, thus only displaying the texts generated by models `bioGPT` and `biobart-large`.

Table B.14: Generation example with positive polarity from USMLE-Symp test set.

Triplet	(47-year-old man, Suffers_from, Polyarteritis nodosa), (47-year-old man, Has_symptom, joint and muscle pain), (47-year-old man, Has_symptom, diffuse abdominal pain), (47-year-old man, Has_symptom, diffuse tenderness), (47-year-old man, Has_symptom, occult blood), (Polyarteritis nodosa, Has_as_symptom, Fever), (Polyarteritis nodosa, Has_as_symptom, Elevated circulating C-reactive protein concentration), (Polyarteritis nodosa, Has_as_symptom, Hypertension), (Polyarteritis nodosa, Has_as_symptom, Abnormality of cardiovascular system morphology), (Polyarteritis nodosa, Has_as_symptom, Raynaud phenomenon), (Polyarteritis nodosa, Has_as_symptom, Cardiomyopathy)
Reference	The patient is showing Polyarteritis nodosa as the following symptoms are direct symptoms of Polyarteritis nodosa and appear in the case description: abdominal pain.
	`bioGPT`
Prefix Tuning	The 47-year-old man has the following symptoms: joint and muscle pain, diffuse abdominal pain, diffuse tenderness, occult blood and Elevated circulating C-reactive protein concentration. The patient has the following symptoms: Raynaud phenomenon, cardiomyopathy and Raynaud phenomenon.
Control Prefixes	These symptoms indicate that the patient my suffer from Polyarteritis nodosa: Joint and muscle pain, Diffuse abdominal pain, Diffuse tenderness, Diffuse tenderness and Elevated circulating C-reactive protein concentration.
Prefix Pooling	The 47-year-old man has been diagnosed with Polyarteritis nodosa with the following symptoms: Joint and muscle pain, diffuse abdominal pain, diffuse tenderness, diffuse tenderness, occult blood, Fever and Elevated circulating C-reactive protein concentration.
	`biobart-large`
Prefix Tuning	The following symptoms indicate that the patient has Polyarteritis nodosa: joint and muscle pain, diffuse abdominal pain, occult blood, Fever, Elevated circulating C-reactive protein concentration and Cardiomyopathy.
Control Prefixes	The patient can be diagnosed with Polyarteritis nodosa as they are showing the following symptoms: Abnormality of cardiovascular system morphology, Raynaud phenomenon and Cardiomyopathy.
Prefix Pooling	The following symptoms, diffuse abdominal pain, diffuse tenderness, occult blood, Fever, Elevated circulating C-reactive protein concentration, Hypertension and Cardiomyopathy, which are direct symptoms of Polyarteritis nodosa.

We achieve excellent performances for shorter and easier triplet sets, while the quality drops quite low on longer "why" and "why-not" explanations. As mentioned before, this is to be attributed to a certain degree of misalignment between the triplets and the provided references and to the fact that we do not have many such complex inputs in the training dataset. From Table B.14 we notice that both Control Prefixes and Prefix Pooling improve over Prefix Tuning for model `bioGPT`. Indeed, they link the symptomatology with a diagnosis, while `bioGPT` in Prefix Tuning setting first correctly enumerates the patient symptoms and then wrongly states that he is also affected by all the characterizing signs of *Polyarteritis Nodosa*. Similarly, Table B.12 shows that Prefix Pooling and Prefix Tuning obtain very similar results, while Control Prefixes using polarity as a guidance signal outperforms them both. In fact, we can see that the verbalizations produced by `biobart-large` and `bioGPT` in Control Prefixes closely resemble the provided gold reference.

Table B.15: Second generation example with positive polarity from the USMLE-Symp test set.

Triplet	(Galactosemia, Has_as_symptom, Delayed speech and language development)
Reference	Delayed speech and language development is a symptom of Galactosemia.
	`bioGPT`
Prefix Tuning	Delayed speech and language development is a symptom of Galactosemia.
Control Prefixes	People with the disease Galactosemia may show Delayed speech and language development as a symptom.
Prefix Pooling	People with the disease Galactosemia may show Delayed speech and language development as a symptom.
	`biobart-large`
Prefix Tuning	People with the disease Galactosemia may show Delayed speech and language development as a symptom.
Control Prefixes	People with the disease Galactosemia may show Delayed speech and language development as a symptom.
Prefix Pooling	Delayed speech and language development is a symptom of Galactosemia.

Appendix C

Visualizations

We now show some interesting visualizations related to the prefix parameters in our models. In particular, in Section C.1 we display low-dimensional representations of both the control prefixes and the components of the prefix pool. This is done in order to verify whether the model is actually able to capture information about the semantic similarity between categories, which we assume when performing zero-shot learning of control prefixes for unseen categories. We also report heatmaps of the attention weights at the last layer of our transformer architectures in Section C.2.

C.1 T-SNE Prefixes Representation

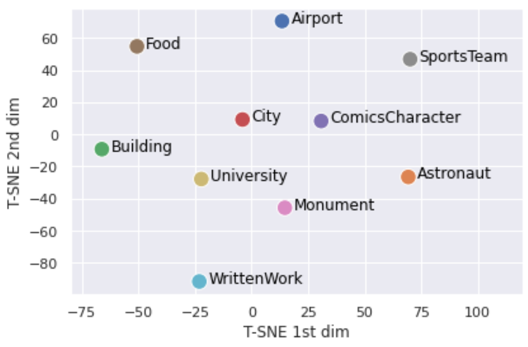
In the following sections, we show low-dimensional representations of the category control prefixes our models learned, both for the general-purpose WebNLG dataset and the medically-oriented USMLE-Symp data. We also display the ten prefixes that make up the pool in Prefix Pooling setting.

t-Distributed Stochastic Neighbor Embedding (t-SNE) [47] is a dimensionality reduction technique that is particularly well suited for visualizing high-dimensional data. Indeed, a dataset may have thousands of dimensions, but often not all these features are necessary to represent the points in a faithful way, making sure that those which are close together remain so and vice-versa. t-SNE models dimensionality reduction as an optimization problem: it first computed the euclidean distances between any two points of the original dataset; then, it uses these distances to generate conditional probabilities that evaluate how likely it is for two points to be neighbors. At this point, a random set of elements in 2 or 3 dimensions is created, and the same conditional probabilities are computed for it as they were computed for the original data. Finally, during the optimization phase, the two probability distributions are brought to be as close as possible one to the other. The most important hyperparameter to set for this technique is the *perplexity*, which closely correlates with the expected number of neighbors: a low value makes the algorithm focus on the closest points, while the higher it is the more t-SNE tries to get a large-scale, big-picture of the data. All our plots are realized with perplexity set to 6 and a maximum number of iterations of 600.

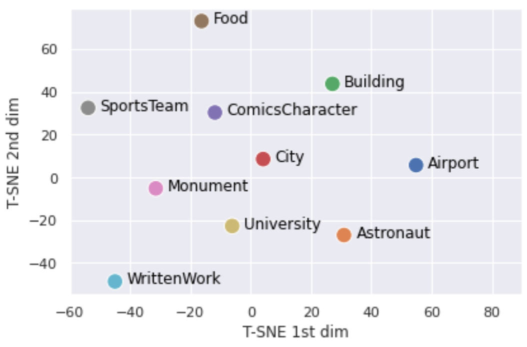
C.1.1 WebNLG Visualizations

We now show t-SNE visualizations of control prefixes and prefix pool learned by models `gpt2-large` and `t5-large`, which achieved particularly good results on the WebNLG dataset.

Control Prefixes Figure C.1 displays masked-self attention components of each control prefix (comprising multiple key-value pairs at each layer) for `gpt2-large` in two different scenarios, with and without layer dependency. We can see that, while the topology of the prefixes undoubtedly changes, some important similarities remain: for instance, *City* is always quite close to *University*, *Airport*, and *Monument*. Similarly, the models both seem to recognize a certain semantic closeness between *WrittenWork* and *University*.



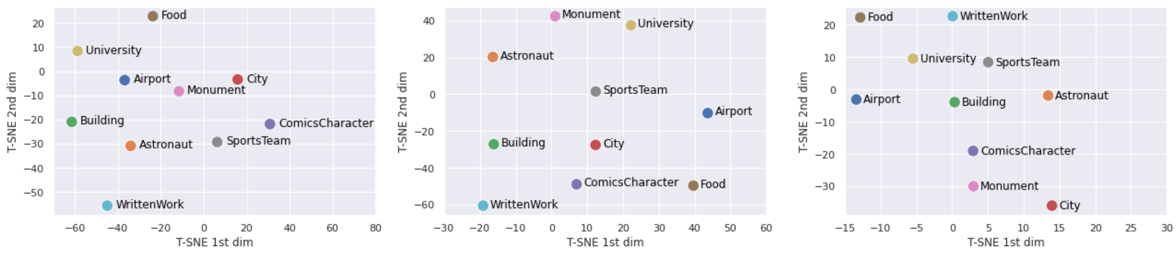
(a) Without layer dependency



(b) With layer dependency

Figure C.1: Control prefixes learned for WebNLG by model `gpt2-large` with and without layer dependency.

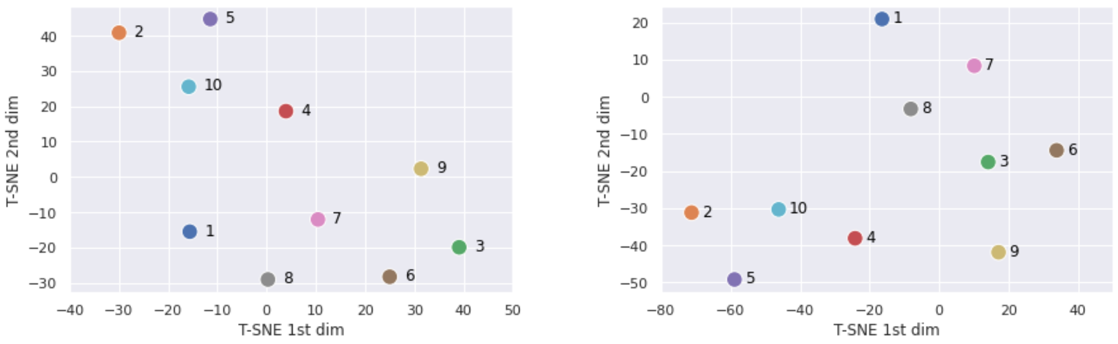
Next, Figure C.2 displays the encoder, decoder, and cross-attention components of the prefixes for model `t5-large`. Notice that the first of these three components is particularly relevant, as the category is a piece of additional information about the input, which shall thus be incorporated in the encoder information. Indeed, we can observe how, in Figure C.2a, classes that are semantically close like *City* and *Monument* are located one nearby the other in the low-dimensional space.



(a) Encoder (b) Decoder (c) Cross

Figure C.2: Control prefixes learned for encoder, decoder, and cross attentions in `t5-large` with layer dependency.

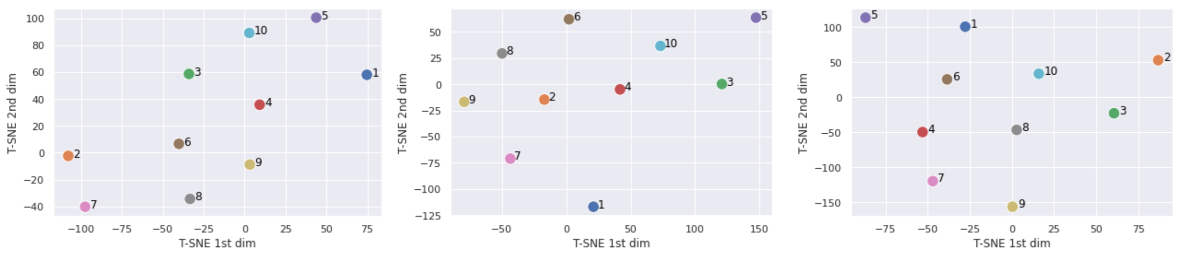
Prefix Pooling Here we first look at the ten prompts learned by our `gpt2-large` model in Prefix Pooling setting, with and without layer dependency. Figure C.3 shows how, despite a clear change in topology, the closeness between some components remains. For instance, prefixes 2 and 5 are always quite close together and the same can be said for 6 and 3. Analyzing the distribution of these prefixes with respect to the



(a) Without layer dependency (b) With layer dependency

Figure C.3: Prefix Pool learned for WebNLG by model `gpt2-large` with and without layer dependency.

category of the validation instances, we indeed see that those with indexes 3 and 6 are quite often used with instances of type *Food* or *Building*, thus their similarity makes sense. Figure C.4 then shows the encoder, decoder, and cross-attention components of the prefix pool for model `t5-large`. We should note that the first two are probably the most significant components, as one should capture eventual additional information about the input, while the other should represent new desired characteristics for the output verbalizations.



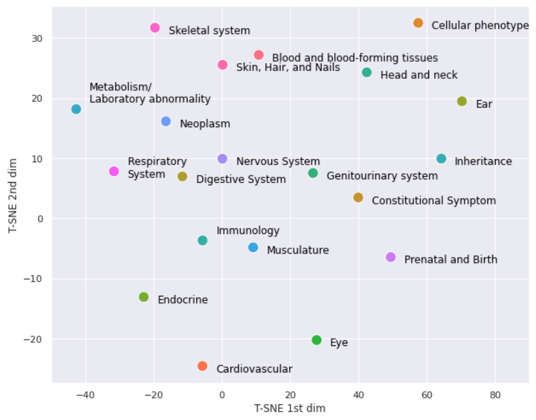
(a) Encoder (b) Decoder (c) Cross

Figure C.4: Prefix pools learned for encoder, decoder, and cross attentions in `t5-large` with layer dependency.

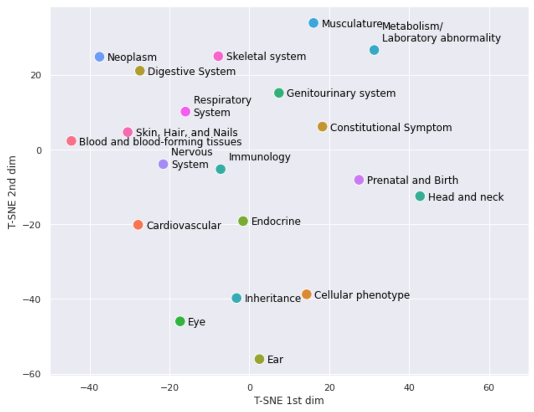
C.1.2 USMLE-Symp Visualizations

We now show t-SNE visualizations of control prefixes and prefix pool learned by two models (`bioGPT` and `bioBART-large`) on the USMLE-Symp dataset. We use these two architectures as they we able to achieve good performances with most settings in the experiments of Chapter 6.

Control Prefixes Regarding the Control Prefixes setting, we should remember that our experiments included three different configurations: polarity-only, category-only, and the combination of polarity and category. We will now only focus on the last two possibilities, as we expect no meaningful non-obvious semantic relationships to emerge from displaying the two polarity control prefixes. Figure C.5 shows the category prefixes learned by `bioGPT` when including and not including polarity as an additional guidance signal. We can notice that, despite changes in the topological placements of the elements, some of them remain close together: for instance, *Skin, Hair and Nails* is always near to *Blood and blood-forming tissues*. Similarly, both models consistently place *Consitutional symptom* close to *Genitourinary system.* Overall, examining the produced visualizations, we notice that the similarities encoded when also taking into account the polarity information seem to be more realistic: this would indeed be in accordance with our results in Section 6.4, where the polarity+category setting achieved better performances.



(a) Without polarity



(b) With polarity

Figure C.5: Control prefixes learned for USMLE-Symp by model `bioGPT`, with and without additional polarity information.

Next, Figure C.6 shows the encoder components of the category control prefixes for our best configuration, that is, the one using polarity and class as two separate guidance signals. We can immediately notice that the two similarities we pointed out before are not as obvious here: this leads us to hypothesize that our models struggle to find actual

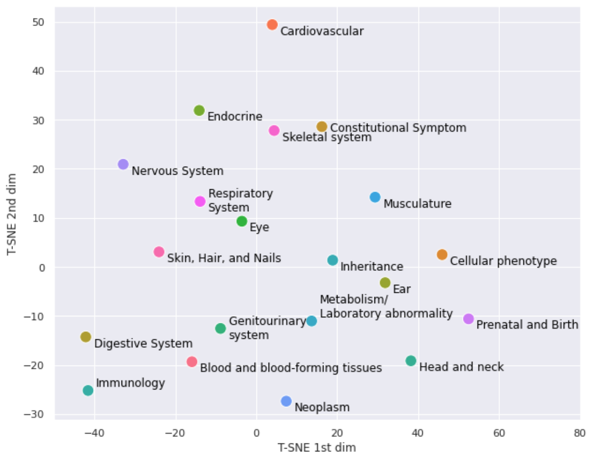


Figure C.6: Encoder control prefix components for `biobart-large` on the USMLE-Symp dataset.

differences between the various semantic classes. Indeed we previously pointed out how category information might be superfluous for the USMLE-Symp data, where most of the triplets have the same structure and the same predicate, no matter their topic.

Prefix Pooling Regarding the Prefix Pooling setting, we now display the ten prompts learned by our `bioGPT` model in Figure C.7, while Figure C.8 shows the encoder, decoder, and cross attentions components of the prefix pool for model `biobart-large`.

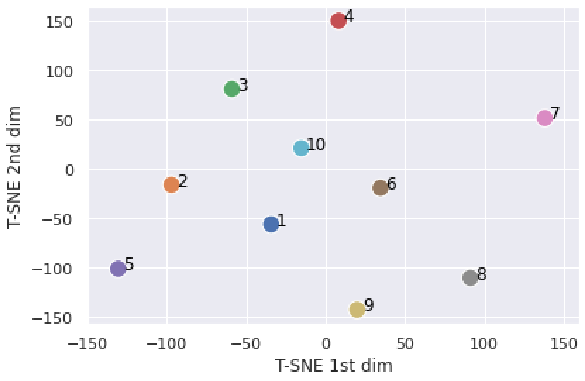
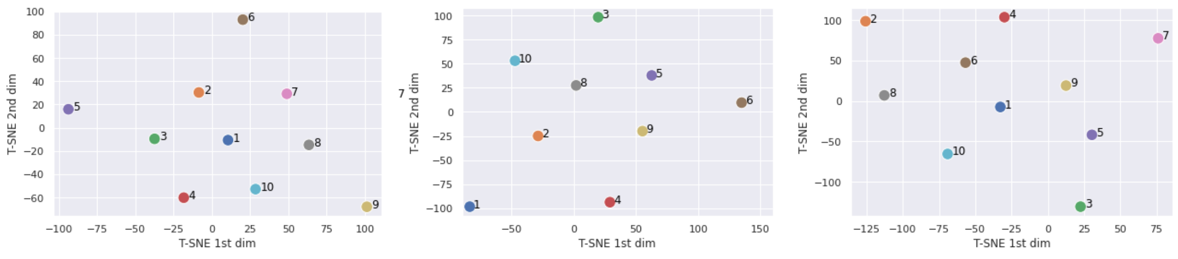


Figure C.7: Prefix pools learned by `bioGPT` on the USMLE-Symp data.



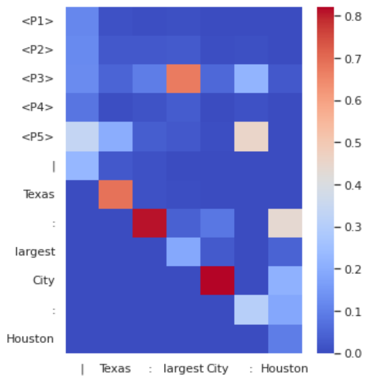
(a) Encoder (b) Decoder (c) Cross

Figure C.8: Prefix pools learned for encoder, decoder, and cross attentions in `biobart-large` on the USMLE-Symp data.

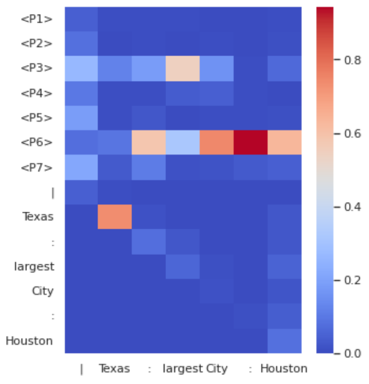
We can immediately see how some prefixes appear close together in both `bioGPT` and the encoder component of `biobart-large`. For instance, prompt 2 is always quite close to prompt 1, as is prompt 10. By looking at how these prompts correlate with the HPO category associated with each input we do indeed see that those with index 1 and 2 are often used fro classes *Endocrine* and/or *Cardiovascular*. Decoder and cross-attention components in `biobart-large` are less easily interpretable.

C.2 Attention Heatmaps

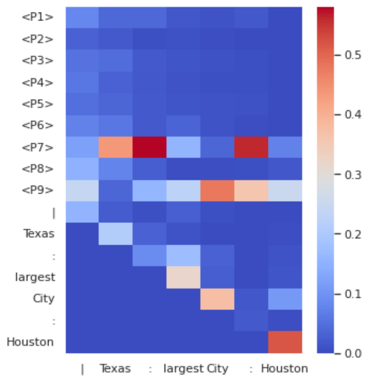
In this final section, we show heatmaps representing the weights for the decoder masked self-attention in two autoregressive models, that is, `gpt2-large` for the WebNLG dataset and `bioGPT` for USMLE-Symp. Figure C.9 shows how these values relative to head 14 of the last transformer layer change for Prefix Tuning, Control Prefixes, and Prefix Pooling. Overall, it seems that the longer the prefix the less important the starting prefix tokens start to be on the overall results. Finally, Figure C.10 shows the attention weights for the last layer of model `bioGPT`; we again choose to display those related to head 14 and compare the Prefix Tuning setting with the two dynamic prefixes approaches.



(a) Prefix Tuning

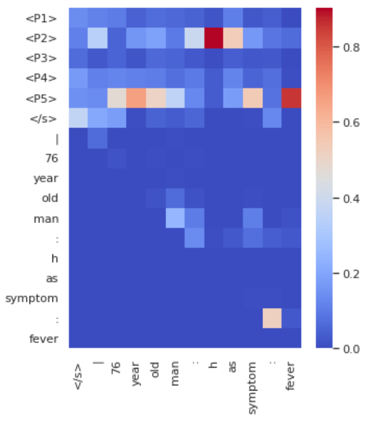


(b) Control Prefixes

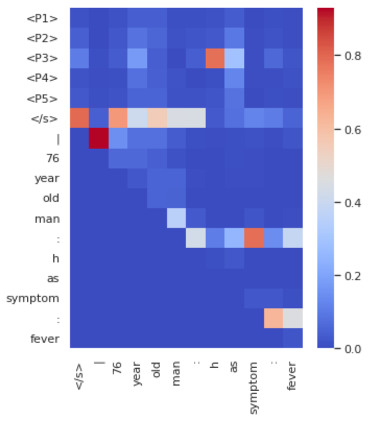


(c) Prefix Pooling

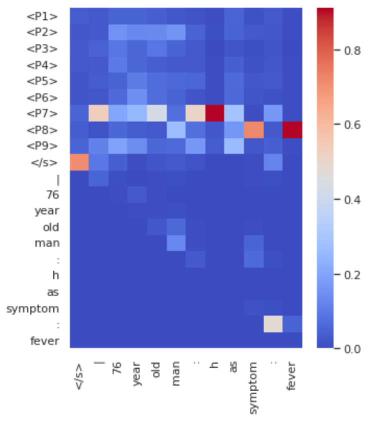
Figure C.9: Comparisons of attention weights of head 14 of the last layer in `gpt2-large` for the WebNLG dataset.



(a) Prefix Tuning



(b) Control Prefixes



(c) Prefix Pooling

Figure C.10: Comparisons of attention weights for head 14 of the last layer in `bioGPT` for the USMLE-Symp dataset.

Acknowledgements

First, I would like to thank professor Torroni for supporting me in this journey and helping set everything up for my internship abroad. I am also extremely grateful to professors Cabrio and Villata from the Côte d'Azur University for giving me the possibility to work on such an interesting research project and for the support they provided through the entire experimentation and writing process. Moreover, I wish to thank the whole Wimmics team at Inria for welcoming me with open arms during my internship. A special mention should go to Benjamin and Santiago, for their work on the ANTIDOTE project and on the USMLE-Symp dataset. Without them this work might not be here.
I am very thankful for all the great lecturers of the master in Artificial Intelligence at the University of Bologna, and especially to professor Stefano Ferretti for encouraging me to pursue a research career and for the amazing opportunities he offered.
I will always be very grateful to my family, my parents and sister Sara, for always supporting my goals and giving me all the resources and encouragement needed to reach them. Finally, a big thank you to my friends, for always being there and willing to listen whenever I needed to walk through a problem with someone, even if they all know next to nothing about AI.