Master Thesis

in

Data Science

A Prompting-based Encoder-Decoder Approach to Intent Recognition and Slot Filling

Panagiotis Tassias

Academic Supervisor: Ion Androutsopoulos

Department of Informatics

Athens University of Economics and Business

Capstone Project Supervisor: Themos Stafylakis

Omilia Ltd

November 2021

Panagiotis Tassias

A Prompting-based Encoder-Decoder Approach to Intent Recognition and Slot Filling

November 2021

Supervisors: Ion Androutsopoulos (AUEB), Themos Stafylakis (Omilia)

Abstract

In recent years, there is an increasing interest in developing advanced conversational agents that facilitate users to accomplish specific goals. Natural Language Understanding (NLU), a subfield of Natural Language Processing, is at the core of these task-oriented dialogue systems. In this thesis, we experimented with different ways of tackling NLU problems, focusing on the sub-tasks of Intent Recognition and Slot Filling. By conducting various experiments on the publicly available ATIS and SNIPS datasets, we confirm that in cases where there is explicit slot label alignment, fine-tuning large Language Models like BERT, seems to be the gold standard approach. However, regarding the Slot Filling problem, in most real-world cases this method is not feasible due to the absence of human-annotated B-I-O tags and it additionally performs poorly in few-shot settings, where there is a limited set of labeled data. In order to overcome these limitations, we propose using an encoder-decoder approach that incorporates the concept of prompting. Specifically, we utilize the T5 Language Model along with natural language templates which the model is prompted to fill in with the relevant information. This method achieves 98% intent accuracy and 95.9% slot micro-F1-score on the SNIPS dataset. More importantly, it provides substantial performance improvements in few-shot settings and displays great adaptability to different intents and domains, when compared to its counterpart that does not embody prompts.

Acknowledgements

I would like to sincerely thank my supervisors Prof. Ion Androutsopoulos and Dr. Themos Stafylakis for their guidance, support and constructive comments throughout the development of this thesis. I would also like to thank Georgia Athanasopoulou, Sofoklis Kakouros and George Mastrapas for their valuable advice and knowledge instillation.

Contents

Abstract iii

Acknowledgements v

1 Introduction 1
- 1.1 Dialogue Systems . . . 1
- 1.2 Natural Language Understanding . . . 1
- 1.3 Current Approaches . . . 2
- 1.4 Limitations of existing methods . . . 3
- 1.5 Proposed Approach . . . 4
- 1.6 Outline . . . 5

2 Background and Related Work 7
- 2.1 Dialogue System Architecture . . . 7
- 2.2 Rule-based Models . . . 8
- 2.3 RNNs and joint models . . . 9
- 2.4 Pre-trained Language Models (LMs) . . . 9
  - 2.4.1 Transformers . . . 10
  - 2.4.2 BERT . . . 11
  - 2.4.3 T5 . . . 12
- 2.5 Prompting . . . 13

3 Methods 17
- 3.1 BERT Fine-Tuning . . . 17
- 3.2 T5 Fine-Tuning . . . 18
- 3.3 T5 Fine-Tuning with Prompting . . . 20
- 3.4 T5 Generalization to Few-Shot Slots . . . 22

4 Experiments 23
- 4.1 Datasets . . . 23
- 4.2 Evaluation Measures . . . 24
- 4.3 Training Details . . . 25
- 4.4 Experimental Results . . . 26

4.5 Ablation Analysis 27

5 Conclusions and Future Work **29**

Bibliography **31**

List of Figures **35**

List of Tables **36**

1 Introduction

1.1 Dialogue Systems

A dialogue system (or conversational agent) is a software system that converses with a human in natural language. Dialogue systems have become an ubiquitous part of our lives and a gold standard in research and industry. These systems have the flexibility to be integrated into all kinds of applications; call centers, websites, personal digital assistants and messaging platforms to name a few. Most individuals use their phone's integrated voice assistant to accomplish all kinds of tasks, like listening to their favorite music or turning on the lights. Moreover, almost every industry nowadays has adopted some kind of conversational agent to facilitate the interaction with their customers. This adoption implies numerous benefits for the corporate world which include reducing human costs and serving bulks of customer requests instantaneously. According to [13], dialogue systems are divided into two main categories; the **task-oriented** and the **non-task-oriented** dialogue systems. The former refers to systems that their end goal is to help the user achieve a specific task, like booking a flight or completing a bank transaction. On the other hand, non-task-oriented systems (also known as chatbots) are developed with focus on mimicking humans in open-domain, end-to-end conversations with no specific goals and are oftentimes used for entertainment purposes. In the present thesis, we will focus on task-oriented systems and more specifically on their sub-task called Natural Language Understanding.

1.2 Natural Language Understanding

A modern dialogue system consists of many different components, which have discrete roles and purposes. All these components collectively are collectively assembled in a pipeline called a dialogue system. A detailed description of these modules will be conducted in the next section. In the context of this writing, we will focus on Natural Language Understanding (NLU), which is a vital component of a conversational agent. As its name suggests, the NLU unit is responsible for understanding the meaning of the user's utterance. A typical NLU module aims at

translating an utterance into a semantic representation and usually is decomposed in three sub-tasks:

- Domain Classification (DC), where the system needs to detect the general subject of the user's query, e.g. "Air Travel",
- Intent Recognition (IR), which classifies the intent of the utterance, e.g. "Book a flight", "Cancel a flight",
- Slot Filling (SF), which performs a sequence tagging procedure, where input tokens are labeled with tags indicating words expressing particular roles (e.g., the origin or destination of a requested flight), thus extracting word-level information with respect to the user's intent.

In this work, we mainly concentrate on the problems of Intent Recognition and Slot Filling, focusing mostly on the latter. Slot Filling is inherently more complex, which makes it more challenging.

Intent recognition can be seen as a multi-class classification problem, where the input utterance should be classified into one intent, from a set of predefined classes. On the other hand, during the slot filling process, each individual input token or word is traditionally mapped to a token label, by utilizing the so-called BIO tags. BIO stands for the words "Begin", "Inside", and "Outside". For example, assuming the input utterance is "Find me flights to New York", the word "New" should be tagged as "Begin-destination_city_name" (or more often "B-destination_city_name") and the word "York" as "Inside-destination_city_name" (or "I-destination_city_name"). This means that the two tokens together comprise a single entity of concern, that is a "destination_city". In addition, all the remaining tokens should be tagged as "Outside" (or "O") as they hold no significant information in this specific context.

1.3 Current Approaches

The problems of Intent Recognition and Slot Filling have been studied thoroughly during the past [20]. Early approaches mainly included rule-based systems, which utilized hand-crafted rules and conditions, in order to capture the intent and fill the relevant slots of an utterance. Later on, sequential neural networks like Recurrent

Neural Networks (RNNs) and their variants (LSTMs, GRUs) became the dominant models, capable of extracting meaningful information from textual sequences. Moreover, taking into consideration the high interdependency between the two problems and the invention of the attention mechanism [3], which enhanced the performance of RNN-based models, previous work proposed joint attention-based models [18, 10]. More recently, these neural network modules were replaced by the pre-trained Language Models [9, 19]. These models produced state-of-the-art results in almost every NLP task and also performed very well in NLU in particular [5]. Finally, with the release of the GPT-3 model [4], a new way of leveraging Language Models emerged namely, prompting. The concept of prompting, attempts to transform every NLP problem to a "Masked Language Modeling" problem, exploiting the unsupervised pre-training procedure Language Models undergo. [8] applies prompting methods to tackle the problems of IR and SF. An extensive analysis of the majority of the aforementioned approaches will be conducted in the following chapter.

1.4 Limitations of existing methods

Despite the existence of a variety of approaches concerning NLU, the field still remains extremely challenging, mainly due to the flexibility of the human language; the vast possibilities of expressing a query in different ways, impose extreme complexity to language understanding. Hence, the approaches discussed above have their own limitations. The currently dominant NLU approach is to fine-tune (further train on task-specific training data) a pre-trained model (e.g., pre-trained with masked language modeling on large corpora). This approach usually requires large volumes of human-labeled data, which is rarely the case in real-world scenarios. That is the reason why there is a huge interest recently in evaluating the different methods in **few-shot** settings (where there is low data availability), in which fine-tuning seems to be under-performing. In addition, in the slot filling case, there is an explicit need for label alignment, meaning that each token should be tagged with its corresponding BIO tag in the training data, as shown in Table 1.1. However, in practice this process requires extensive human annotation and results in extra costs. In most instances, there is a human agent who registers only the relevant slot fillers as they have been extracted by the user utterance, producing a kind of weakly labeled dataset, in the sense that there is **no explicit alignment** between the query and the slot tags, as illustrated in Table 1.2. Moreover, there are cases where the slot filler of concern is not explicitly mentioned inside the utterance. Inspecting again the example of Table 1.2, although the user mentions

the cities “Toronto” and “Big Apple”, we are interested in transforming these tags to the code names “Trn” and “NY”. To the best of our knowledge, little effort in the relevant literature has been put in addressing the issue of the absence of slot label alignment.

Utterance	Find	me	flights	from	Toronto	to	Big	Apple
BIO Tags	O	O	O	O	B-fromloc	O	B-toloc	I-toloc

Tab. 1.1: An example of slot annotation with the usage of BIO tags.

Utterance	Find me flights from Toronto to Big Apple
Slots	fromloc: Trn, toloc: NY

Tab. 1.2: An example of slot annotation where there is no alignment between the input utterance and the slot labels. Additionally, the slot labels do not necessarily consist of an input word or span.

1.5 Proposed Approach

Following the work of Chen et al. [5], we initially implement a BERT model, fine-tuned on the publicly available ATIS and SNIPS datasets [11, 7], to jointly model both the intent and the slots of an utterance. This approach benefits from the alignment present in these datasets and operates as a strong baseline for further experiments. In the sequel, taking into consideration the limitations of the existing techniques, we propose using a sequence-to-sequence model, in our case the T5 model [21]. Using an encoder-decoder architecture, sequence-to-sequence models are capable of producing outputs of various lengths which renders them suitable for cases where there is lack of explicit alignment in the slot labels. Furthermore, with regard to the restrictions imposed by the small size of the annotated datasets, we utilize the concept of prompting and transform the IR and SF problems into a kind of masked language modeling one. This experiment is conducted using the SNIPS dataset. To sum up, regarding the slot filling sub-task of NLU, our contributions are two-fold:

- we transform the problem into a text-to-text scheme, by utilizing a sequence-to-sequence Language Model (T5) in order to tackle the problem of lack of slot label alignment,
- we examine utilizing the concept of prompting along with T5, in low data regimes.

Furthermore, the main findings of this thesis are:

- fine-tuning BERT is a dominant approach when slot label alignment is present,
- in the absence of slot label alignment, modern sequence-to-sequence Language Models constitute a decent workaround in reformulating slot filling as a text-to-text problem,
- in few-shot settings, prompting techniques provide substantial performance improvements to slot filling.

1.6 Outline

The remaining of the thesis is organized as follows:

Chapter 2

Provides background information and discusses previous work related to the topic of this thesis.

Chapter 3

Analyzes the proposed methods.

Chapter 4

Provides details regarding the datasets, the evaluation measures and the experimental results.

Chapter 5

Contains the conclusions and possible future work.

2 Background and Related Work

2.1 Dialogue System Architecture

As depicted in Figure 2.1, a modern dialogue system consists of several components, namely:

- the *Automatic Speech Recognition (ASR)* module, which transforms the user's speech signal into textual format,
- the *Natural Language Understanding (NLU)* module containing the Domain Classification, Intent Recognition and Slot Filling procedures, which are the focal points of this writing. This component processes the text obtained by the ASR module and produces the so-called Semantic Frame of the query, holding the relevant information of the turn,
- the *Dialogue Manager*, which consists of the *Dialogue State Tracker (DST)* and the *Dialogue Policy*. DST is responsible for tracking the current state of the conversation including the dialogue history as well as the slot-fillers defined by the user. Dialogue Policy refers to the decisions the system makes, regarding the next action/response to the user,
- the *Natural Language Generation (NLG)* component, which produces the text response in natural language, taking into account the output of the Dialogue Policy sub-module and
- optionally, if the system supports voice-based replies, a *Text-to-Speech (TTS)* module is invoked, that converts the textual output of NLG into a vocal message.

In the following sections, we will analyze the different strategies adopted in relevant bibliography to confront the challenges opposed by NLU problems. Specifically, we will present them in chronological order.

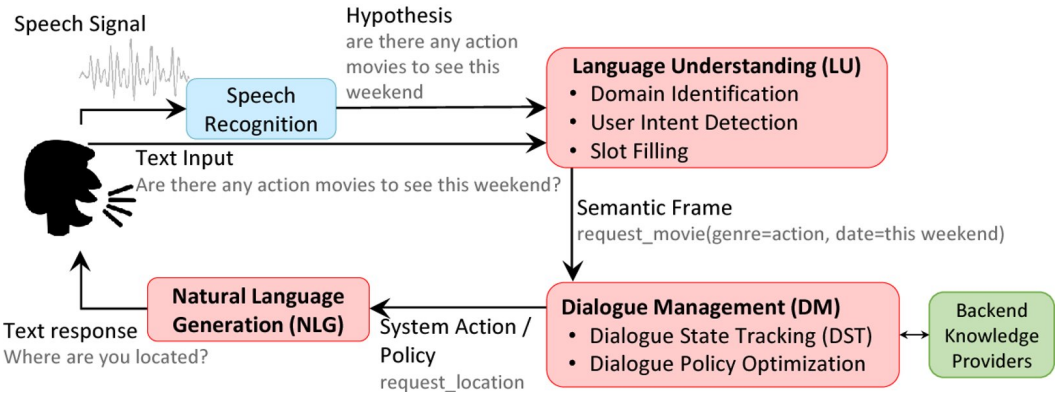

Fig. 2.1: Components of a modern Dialogue System [6].

2.2 Rule-based Models

In these early methods, the system developer was responsible for identifying the underlying patterns that comprise an intent and its corresponding slots. As depicted in Figure 2.2, this was achieved by using regular expressions or, more generally, grammars. Hand-written conditions attempted to capture the semantics of an utterance by matching specific hard-coded keywords to intents and entities of concern, often requiring prior domain knowledge.

SHOW	→	show me | i want | can i see|...
DEPART_TIME_RANGE	→	(after|around|before) HOUR | morning | afternoon | evening
HOUR	→	one|two|three|four...|twelve (AMPM)
FLIGHTS	→	(a) flight | flights
AMPM	→	am | pm
ORIGIN	→	from CITY
DESTINATION	→	to CITY
CITY	→	Boston | San Francisco | Denver | Washington

Fig. 2.2: Rule-based systems [13].

2.3 RNNs and joint models

When machine learning and especially deep learning techniques emerged, they led to substantial progress in the development of a wide variety of Natural Language Processing (NLP) applications, including NLU tasks, thus rendering data-driven approaches the new standard. More specifically, RNNs, particularly LSTMs and GRUs became the new standard for processing text sequences. The majority of these methods tackled IR and SF as separate problems [26, 14]. Later on, approaches that addressed the two problems jointly emerged. These methods train a single system on both tasks, thus reducing the computational and storage overhead while at the same time exploiting the interdependence between the two problems. [18] follows such an approach and constructs a joint model adopting a bidirectional RNN encoder-decoder framework, while taking advantage of the attention mechanism. It is also referring to the problem of the absence of slot alignment and in order to address this, it incorporates attention to help the model learn a soft alignment and decode at the same time. Indicatively, the F1-score of the proposed method drops from 95% to 81% when alignment is not present in the ATIS dataset. Based on this work, [10] introduced a slot-gated mechanism to explicitly model the relationship between the intents and the slots and achieved state-of-the-art results. However, lately these approaches are dethroned by the so-called Pre-trained Language Models.

2.4 Pre-trained Language Models (LMs)

Language Models was initially a term referring to models that had the sole purpose of evaluating the fluency of a natural language text, however lately they demonstrate great capabilities in text understanding and in incorporating common knowledge [12]. In order to address the absence of human-annotated data for most NLP applications, there is a recent shift in research towards exploiting the abundance of publicly available unlabeled text. Thus, the concept of transfer learning is utilized and a variety of pre-trained Language Models emerged. The term pre-trained refers to the unsupervised procedure these systems undergo, which includes training on different objectives on huge text corpora like Wikipedia. Typically these LMs consist of millions or even billions of learnt parameters which ideally embody broad language knowledge, hence the name Language Models. Most contemporary approaches using Language Models, apply the fine-tuning paradigm, that is appending a task-specific head (e.g. a Fully Connected Layer or

MLP) for the downstream task of concern and jointly fine-tune (further train) the weights of the pre-trained LM and learn the weights of the task-specific head.

2.4.1 Transformers

Transformers, introduced by [25], have led to huge performance improvements in a variety of NLP tasks. As shown in Figure 2.3 [1], they comprise an encoder-decoder architecture. Models with this type of structure have traditionally been used for tasks like machine translation, summarization and text generation. They are capable of producing sequences of various lengths rendering them more flexible.

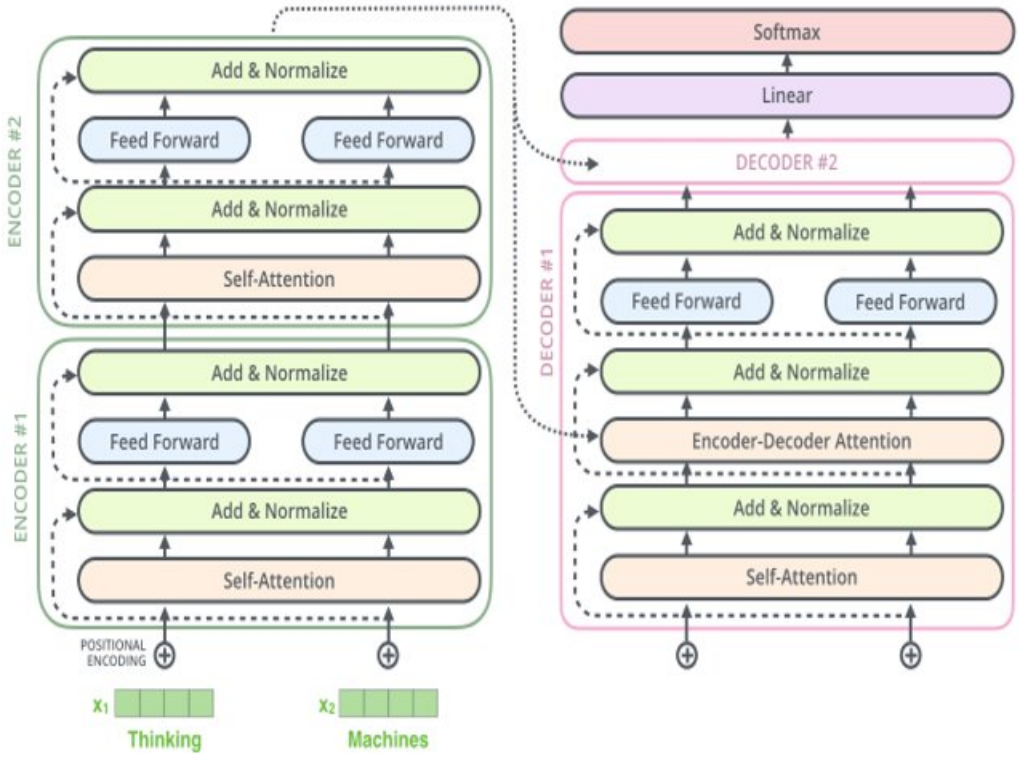

Fig. 2.3: A Transformer consisting of 2 stacked encoders and decoders.

They are based solely on the attention mechanism and a few Feed-Forward and Layer Normalization blocks. They replace the recurrent architecture of earlier sequential neural networks with the so-called residual attention blocks, while still being able to handle long-range dependencies efficiently. More importantly, the novelty of their structure is that these residual attention blocks they consist of, allow high parallelization, which implies dramatically reduced computational needs and increased training speeds. Transformers constitute the gold standard in the development of the majority of modern Language Models.

[1] https://jalammar.github.io/illustrated-transformer/

2.4.2 BERT

BERT [9] stands for Bidirectional Encoder Representation for Transformers and since its release in 2018 has achieved significant results on multiple downstream tasks. The backbone of BERT is a stack of bidirectional Transformer layers [25]. In particular BERT-base, which is the model of choice for our experiments, uses a stack of 12 encoder transformer blocks. In essence, BERT learns context-aware representations of the input sentences or tokens taking into account both the left and the right context. It is pre-trained with two main objectives in mind; the first is Masked Language Modeling and refers to the procedure where 15% of the input tokens are "masked" and the model attempts to predict these missing tokens, as depicted in Figure 2.4 [2]. A special [MASK] token is used for representing these tokens. The second objective is called Next Sentence Prediction and is a binary classification task. As its name suggests, given a pair of sentences, the model decides whether the second sentence is actually the sentence that follows the first one. Finally, it is important to note that BERT is pre-trained on BooksCorpus (800 million words) and Wikipedia (2.5 billion words). [5] utilizes the BERT model to cope with the Intent Recognition and Slot Filling problems. Our first approach is based on this work.

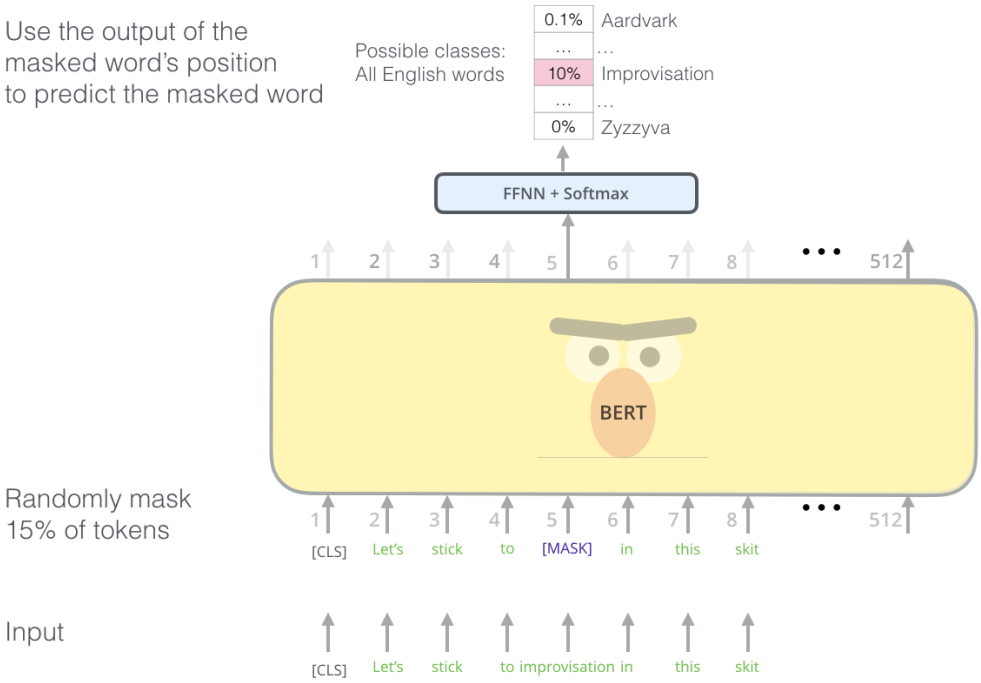

Fig. 2.4: BERT mask-filling pre-training objective.

[2] https://jalammar.github.io/illustrated-bert/

2.4.3 T5

T5 stands for Text-to-Text Transfer Transformer and was released by [21] in 2019. It constitutes a generative Language Model which embeds a sequence-to-sequence architecture. Models with this type of structure have traditionally been used for tasks like machine translation, summarization and text generation. They are capable of producing sequences of various lengths, rendering them more flexible. T5 is also a Transformer-based model, but unlike BERT it uses both encoder and decoder Transformer blocks, like the original Transformer implementation. Its base model is composed of 12 encoder and 12 decoder blocks.

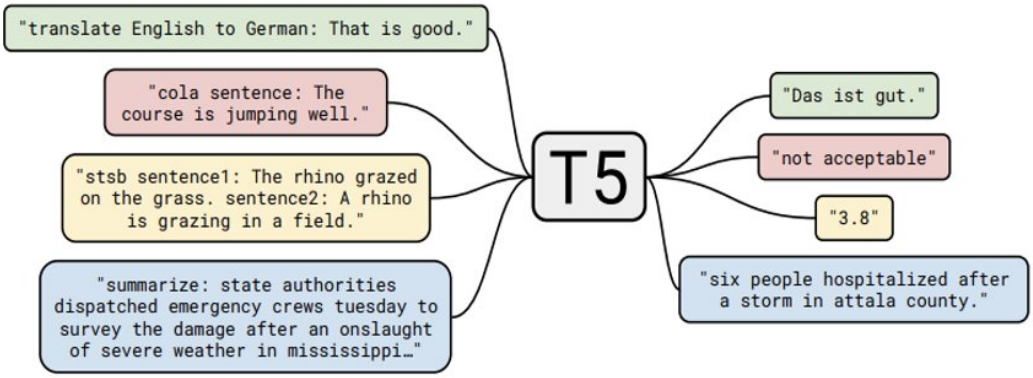

Fig. 2.5: T5 framing various NLP tasks in a text-to-text scheme [21].

T5's authors attempt to formulate every NLP task as a text-to-text problem as Figure 2.5 shows. For instance, to classify a sentence with regards to its sentiment, instead of outputting +1 and -1 for positive and negative sentiment respectively, it outputs the text "positive" or "negative". In addition, T5 supports task-specific prefixes that can be prepended to the input text, such as "classify sentiment: " or "summarize: " leading to performance boost.

Different unsupervised pre-training "denoising" objectives were tried out, with the most notable and efficient being the *span corruption* depicted in Figure 2.6. This method closely resembles the masking objective of BERT, except that in this case whole spans of various lengths are masked out and the model is challenged to unveil them correctly. T5 enumerates the missing spans in a sequence, by utilizing special tokens, called sentinel tokens, shown as <X>, <Y>, <Z> in Figure 2.6. T5 actually calls the sentinel tokens <extra_id_x> where x ranges from zero up to 100. T5 is pre-trained in an enormous custom dataset scrapped by the authors, named Colossal Clean Crawled Corpus (C4), which contains about 750 gigabytes of text.

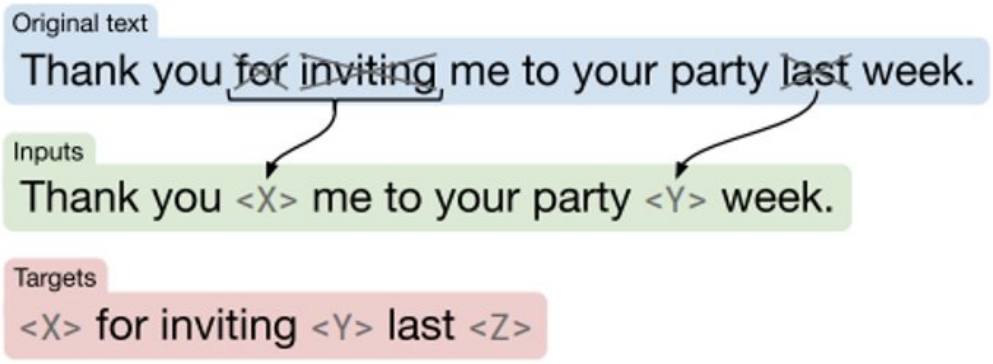

Fig. 2.6: T5's span corruption pre-training objective [21].

[1] proposed using T5 to tackle the problems of Intent Classification and Slot Filling in a sequence-to-sequence manner. Although it achieves remarkable results, it is limited in searching specific input spans, whereas we are interested in providing the flexibility to output values not explicitly mentioned in the input utterance. Referring to the example given in the previous section (Table 1.2), the user may mention the city "Big Apple", while the corresponding slot label of concern is "New York" or merely "NY".

2.5 Prompting

After the recent work of Brown et al. [4] that was accompanied by the release of the GPT-3 model, there is an increasing interest in a new paradigm called prompting. Prompting is one of the main inspirations of this thesis. The reasoning behind this new concept is simple yet powerful. It aims at taking full advantage of the pre-training objective of masked language modeling and aligning every NLP task with this objective. It follows a kind of fill-in-the-blanks scheme by inserting a few extra tokens to the input sequence, often referred to as template, along with one or more mask tokens and asks the model to fill these blanks (mask tokens) with natural language. Hence, in its simplest form prompting does not add new model parameters, in contrast to traditional fine-tuning methods. As a simple binary classification example, assume that we have a sentiment classification problem in which we would like to classify the movie review *"Don't waste your time watching it"*. In this case, a template containing a mask token can be appended to the input resulting in the phrase: *"Don't waste your time watching it. It was [mask]"*. It is natural to expect from the model to produce a higher probability for the word "terrible" than "great" for filling the mask token, thus indirectly classifying the

sequence to the negative sentiment, by using these two label words, as they are called, to map each class to a word.

What is important about prompting is the fact that it displays great performance in the so-called few-shot settings, namely in situations, where there is limited data availability for a downstream task, or even in zero-shot settings, where no training data is available. The extensive analysis of [15] concludes that a prompt is "worth" hundreds of training examples highlighting the impact of this method.

After GPT-3, the predecessor of prompting, the PET papers [22, 23] proposed two radical changes that inspired many of the following research papers. Firstly, the authors of these papers decided to use bidirectional LMs, like BERT and its variants, in contrast to GPT-3 which constitutes a unidirectional autoregressive model. This facilitates the procedure of prompting, by allowing the model to attend to the full sentence context when filling a mask. Secondly, they proposed to further fine-tune the LM, as opposed to GPT-3 where the model parameters are kept frozen, achieving significant results, especially in few-shot settings. [2] provides further evidence that prompt-based fine-tuning, where the weights of the LM are fine-tuned along with the prompt tokens, outperforms standard fine-tuning in the full data regime. In addition, a competitive model can be constructed in low-resource settings, by only tuning the biases of the model.

Taking into account the benefits of prompting, it is worth investigating the different ways of designing a prompt. The aforementioned bibliography uses manually designed prompts, by incorporating domain knowledge and commonsense. [12] utilizes text mining and paraphrasing to construct a prompt that helps in mining factual knowledge. The authors also state that prompt ensembling, i.e. training different prompts and utilizing them in a majority voting manner, leads to performance improvements. [24] follows a different approach by training a model to automatically generate prompts that contain words from the vocabulary and achieves remarkable results in a variety of tasks. However, the resulting prompt is not in natural language, which heavily undermines the explainability part of the procedure.

A few following papers use *soft prompts*, i.e. continuous vectors learnt through backpropagation. The main idea of this approach is to add some extra tokens in the input sequence and train the embeddings of these pseudo-tokens, while keeping the rest of the model frozen. The initialization of these token embeddings is either random or based on some prior hand-crafted prompt which uses words from the

LM's vocabulary. [27] follows such an approach for knowledge probing tasks, pointing out that manual prompt initialization provides a good prior for better optimized soft prompts. [17] proposes "prefix tuning" which prepends a sequence of continuous prefixes at every transformer layer and demonstrates competitive results on generative tasks. In essence, the model learns transformer activations that are kept fixed across examples at every network layer. [16] simplifies this idea by only prepending a few tunable tokens to the input text, while keeping the rest of the model parameters fixed and accomplishes on par performance with the traditional fine-tuning method on a variety of NLU datasets. This approach uses the T5 model and in essence, expands the vocabulary of the model with the addition of some learnt pseudo-tokens.

It is obvious that prompting has been examined in a variety of tasks in recent literature. However, little work has been conducted regarding the adoption of prompting techniques in NLU. To the best of our knowledge [8] is the only work following a relevant approach for Named Entity Recognition (NER) problems, with emphasis in few-shot settings. It utilizes a sequence-to-sequence model, in particular BART and adopts the prompting paradigm by providing different templates for the model to complete. Even though it achieves great performance, it follows a kind of counterintuitive method, as it is inspecting consecutive n-grams of the input text, until finding the entity of concern. This requires many extra forward passes for the model, leading to increased computational complexity, whereas our approach outputs the relevant slot fillers in a single run.

3 Methods

3.1 BERT Fine-Tuning

Following the line of [5], as a first step we fine-tune the BERT-base model to jointly model IR and SF, experimenting on both the ATIS and the SNIPS datasets.

For the IR case, we utilize the first special token of BERT, i.e. the *[CLS] token* as shown in Figure 3.1. The output embedding corresponding to this token contains a representation vector for the whole sequence. Hence, it can be used for sequence classification purposes following the traditional fine-tuning approach. We put a classification head on top of that representation, in our case a Linear Fully Connected (FC) Layer, followed by a Dropout layer. Formally, for a given input text $\mathbf{x} = (x_1, \ldots, x_N)$, let $\mathbf{c} \in R^{d_{BERT}}$ be the output vector of BERT corresponding to the [CLS] token, where $d_{BERT} = 768$ is the hidden dimensionality of BERT-base. The intent prediction is obtained as follows:

$$\mathbf{y^c} = softmax(W^c\mathbf{c} + \mathbf{b^c}),$$

where $\mathbf{y^c} \in R^{\#intents}, W_c \in R^{\#intents, d_{BERT}}, \mathbf{b_c} \in R^{\#intents}$. The *Cross-entropy* loss is utilized to compare the predicted distribution with the gold one-hot label vector denoted $\mathbf{l}$:

$$L_I = -\sum_{i=1}^{\#intents} l_i \log \mathbf{y^c}_{(i)}$$

For the SF problem, we utilize the BIO tags as described in the introduction. Similarly to the IR case, we obtain a context-aware representation produced by the BERT module for each individual token and insert a task-specific classifier (again a FC layer) on top of each token embedding, followed by a Dropout layer. Formally, given the input sequence $\mathbf{x} = (x_1, \ldots, x_N)$, let $\mathbf{T} = (\mathbf{t_1}, \ldots, \mathbf{t_N})$ be the context-aware hidden states of BERT corresponding to the input tokens. For a given token i, we can calculate the probability distribution related to the slot classification as:

$$\mathbf{y^s}_{(i)} = softmax(W^s \mathbf{t_i} + \mathbf{b^s}),$$

where $\mathbf{y^s}_{(i)} \in R^{\#slots \times 2+1}, W^s \in R^{\#slots \times 2+1, d_{BERT}}, \mathbf{b}^s \in R^{\#slots \times 2+1}$. The dimensionality of $\mathbf{y^s}_{(i)}$ is equal to $\#slots \times 2 + 1$ because of the "Begin" and "Inside" tags which double the initial number of slots, plus one for the "Outside" case. As in the intent detection case, we utilize the cross-entropy loss to compare the predictions with the true labels. We jointly update the parameters of both the task-specific classifiers and the BERT module itself, based on the sum of the losses of the two problems.

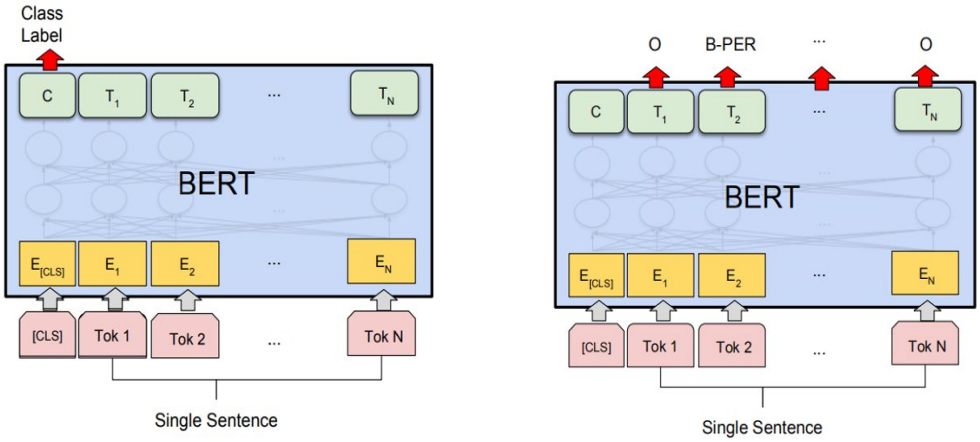

Fig. 3.1: BERT fine-tuning for classification and NER tasks [9].

3.2 T5 Fine-Tuning

For methods implemented from here on, we separate the problems of IR and SF and deal with them independently. The reason is that intent classification constitutes a simpler problem and we would like to pay special attention to the more challenging slot filling process. Moreover, we will concentrate on the SNIPS dataset, which is multi-domain and contains richer vocabulary with more discrete intents and slot values. Finally, we should note that during the sequence-to-sequence experiments, the T5-small variant of T5 is utilized, since the base model is much bigger and is highly demanding with regards to computational power and time. Table 3.1 compares the backbone structures of BERT-base and T5-small, which are the models of choice for the experiments of the present thesis. It can be seen that T5-small contains almost half of the BERT-base weights.

We initially focus on predicting the intent of the utterance. As opposed to the BERT procedure described above, we are no longer able to use discrete integer labels to

	BERT	T5-small
#Vocab	30000	32000
Embedding dim	768	512
Attention heads	12	8
#Layers	12	12
#Params	110m	60m

Tab. 3.1: Underlying architecture comparison between BERT and T5-small.

classify the intents, as T5 is a text-to-text model, thus we should formulate the problem as such. A simple solution to this is letting the model produce a description of the intent in natural language. In other words, given the query of the user as input to the model, we expect a text describing the label to be generated. Hence, we map each prior intent label to a natural language counterpart, which will be easier for the model to understand. For instance, the intent "AddToPlaylist" is mapped to "playlist" and "SearchScreeningEvent" to the simpler "movie" keyword. To give an example from the SNIPS training set, supposing that the utterance is *"give me a list of movie times for films in the area"*, T5 is expected to output the word *"movie"*.

Regarding the slot filling problem, we are concerned about the case in which there is no explicit alignment between the tokens and their corresponding slot BIO tags. As an example we will use the query *"please add iris dement to my playlist this is selena."*, where we would like to obtain the slot information *{"artist": "iris dement", "playlist owner": "my", "playlist": "this is selena"}*. Hence, we transform the dataset accordingly by removing the BIO labels. As an intermediate step, because the problem of intent classification is relatively trivial, we assume that there is an oracle for predicting the intent and thus, it is considered known in advance. In the sequel, we map each intent to its corresponding slots as the training dataset dictates. For instance, among the total of 39 slots (after removing the BIO tags), the intent "AddToPlaylist" is related to only 5 of them, i.e. "artist", "entity_name", "music_item", "playlist" and "playlist_owner". We sort these slots out in alphabetical order and ask T5 to produce a value for each one of those. If the slot filler of concern is present in the input utterance, we expect the model to locate it and put it in the correct slot. Otherwise, it should produce the predefined word "none", which represents the absence of that slot. To achieve this, we leverage the special mask tokens used during the pre-training procedure of T5, i.e. the "<extra_id_x>" sentinel tokens as described in the previous chapter, by inserting one mask token for each relevant slot. Referring again to the previous example, Figure 3.2 depicts the input to the model, as well as the desired output.

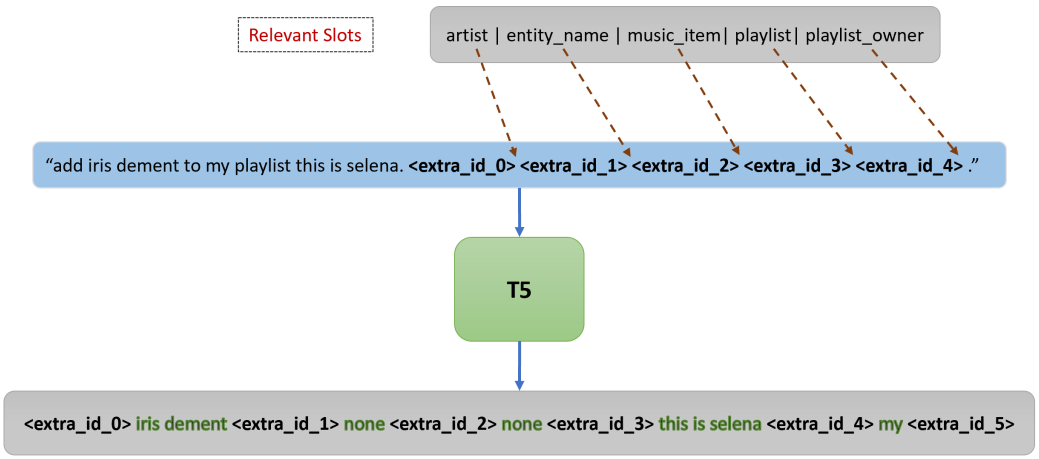

Fig. 3.2: Slot Filling approach using the T5 model. The model is expected to output a text for each relevant slot (with regards to the intent of the input). The keyword "none" is produced when the corresponding slot filler is not present in the utterance.

Giving a high level formulation of the training procedure, let $\mathbf{x} = (x_1, \ldots, x_N)$ be the input sequence of the model. We feed this input to the encoder part of T5 and we obtain the hidden representation $\mathbf{h}^{enc} = encoder(\mathbf{x})$. At the $i-th$ step of the decoding procedure, we obtain the hidden representation $\mathbf{h}_i^{dec} = decoder(\mathbf{h}^{enc}, \mathbf{o}_{1:i-1})$, where $\mathbf{o}_{1:i-1}$ are the previous output words of the model. The probability distribution of the word o_i is:

$$\mathbf{t_i} = p(\mathbf{o}_i \mid \mathbf{o}_{1:i-1}, \mathbf{x}) = softmax(W\mathbf{h}_i^{dec} + \mathbf{b}),$$

where $\mathbf{t_i} \in R^{|V|}, W \in R^{|V|,d_{T5}}, \mathbf{b} \in R^{|V|}$, with $\mid V \mid = 32000$ and $d_{T5} = 512$. The training objective is the minimization of the *Cross-entropy loss* of the prediction tokens compared with the label tokens denoted as $L = (\mathbf{l_1}, \ldots, \mathbf{l_m})$, where $\mathbf{l_i} \in R^{|V|}, i = 1, \ldots, m$:

$$L = -\sum_{i=1}^{m} \sum_{j=1}^{|V|} l_{ij} log t_{ij}$$

3.3 T5 Fine-Tuning with Prompting

Inspired by the paradigm of prompting and its success in low-resource settings, we are also interested in investigating its impact in few-shot scenarios. For that reason, we obtain different slices of the SNIPS dataset. In particular, from the total of about 13K training examples, we take small batches of 200 and 500 training examples at random, preserving the same distribution of intents and slots as the

original full dataset. In the sequel, in order to adopt the concept of prompting, we modify the previous problem formulation with the addition of a hand-crafted template dictating the kind of information we expect the model to generate at each point.

Regarding the intent classification case, a template phrase is appended to the input utterance, along with the special mask token that should be filled. The phrase of choice is *"This sentence refers to "*, so in the case of the previous example the full model input text would be *"please add iris dement to my playlist this is selena. This sentence refers to <extra_id_0> "* and the model should generate the word *"playlist"* which is mapped to the original "AddToPlaylist" intent class.

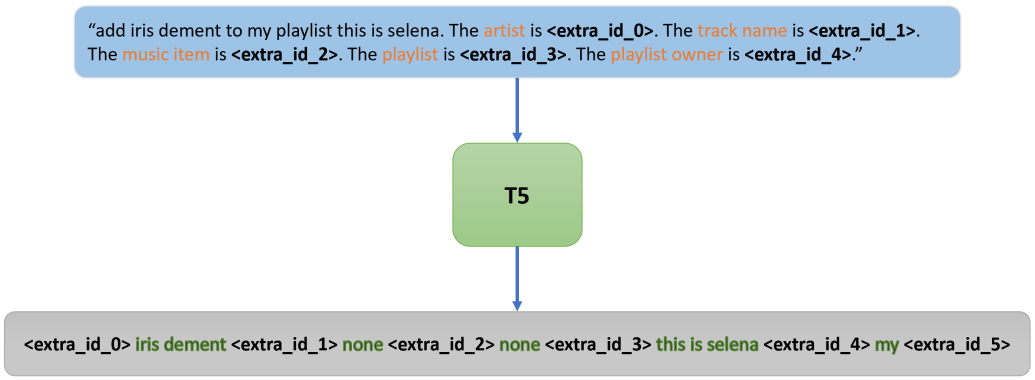

Fig. 3.3: Slot Filling approach using the T5 model along with templates. The model is guided to produce the slot filler of concern with the utilization of templates.

Following the same reasoning for the slot filling problem, having the prior knowledge of the correct intent, we construct a template that attempts to prompt-out all the relevant slot values. This is achieved by providing a template phrase for each individual slot and concatenating all these phrases at the end of the input sequence. Additionally, in order to assist the model in exploiting its internal knowledge of the language, we transform the slot labels into "natural" language terms. That is after removing the underscore symbol where needed, we assign each slot to a word or sequence of words that closely resemble the true slot semantics. For instance, the slot "entity_name" refers to the name of a song, thus it is mapped to "track name". Furthermore, each mapped sequence is surrounded by the phrase *"The [slot_name] is [mask]"*. Figure 3.3 illustrates this formulation, using again the same example as before.The training procedure is exactly the same as in the previous method where no prompting techniques were utilized.

3.4 T5 Generalization to Few-Shot Slots

Relevant bibliography contends that utilizing the concept of prompting results in better performance for unseen intent and slot labels. To validate this claim we perform another experiment. In this scenario, we consider the full training set except for the slots referring to 2 of the total of 7 intents (different pairs were tried out), which were kept few-shot. For those two intents only 40 training examples were used for model training and the evaluation measures are reported only for the subset of the test set referring to those intents.

4 Experiments

4.1 Datasets

- The **ATIS (Airline Travel Information Systems)** [11] dataset consists of manual transcripts from audio recordings of people asking for flight information. It is one of the most used benchmarks for NLU and SLU (Spoken Language Understanding) systems and contains simple single turn utterances. It incorporates 21 different intents and 120 slot labels, but as shown in Figure 4.1 the dataset is extremely imbalanced, with over 70% of the intents being a flight search.

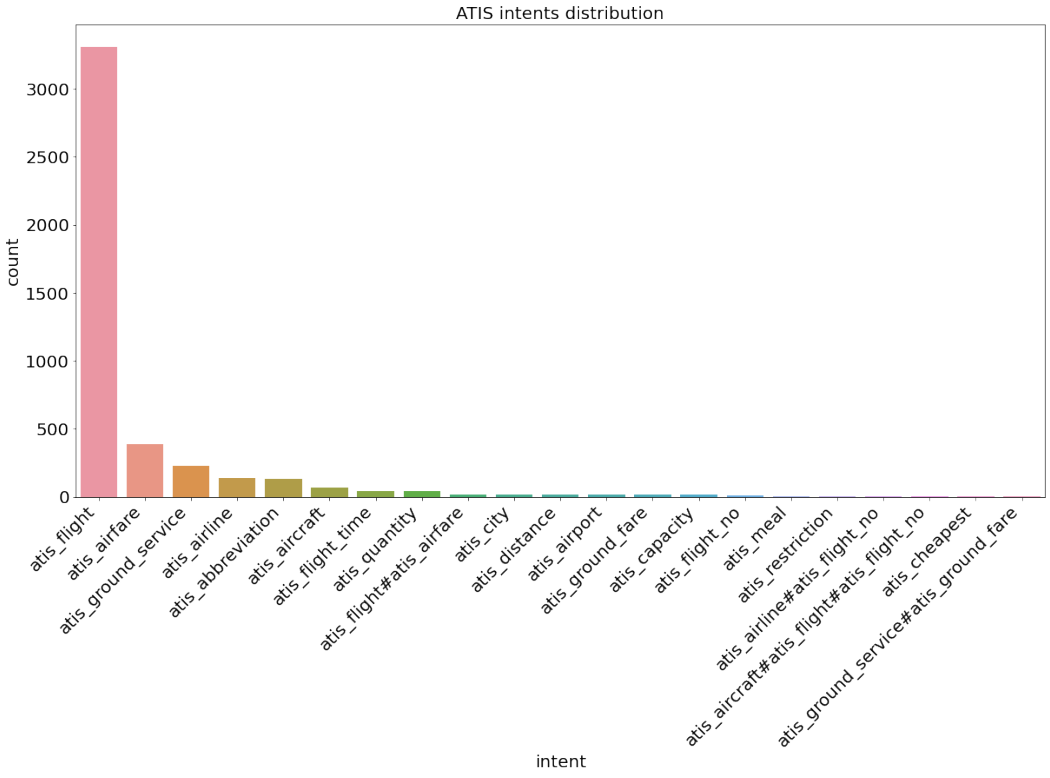

Fig. 4.1: ATIS intent distribution.

- The **SNIPS** [7] dataset embodies queries from the users of the Snips voice platform. It comprises a multi-domain dataset, with intents varying from requesting information related to the weather condition, to finding a restaurant or playing a song. In particular, it contains 7 different intent types and 72 slot labels. As opposed to the ATIS dataset, SNIPS is a balanced dataset in terms

of the intents' distribution and has about three times more training examples. Moreover, due to the diversity in the topics of the queries, it consists of a much richer vocabulary. Table 4.1, summarizes the main statistics of the two datasets.

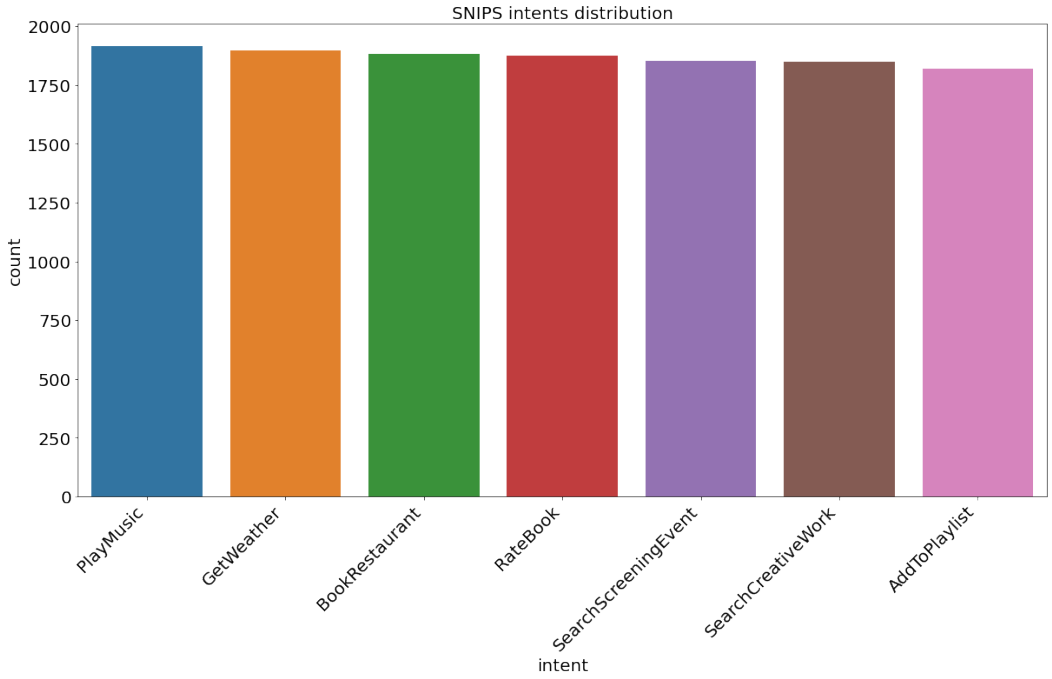

Fig. 4.2: SNIPS intent distribution.

	ATIS	**Snips**
Vocabulary Size	722	11,241
#Slots	120	72
#Intents	21	7
Training Set Size	4,478	13,084
Development Set Size	500	700
Testing Set Size	893	700

Tab. 4.1: Datasets statistics.

4.2 Evaluation Measures

All the different models and configurations are evaluated with respect to their performance over the test datasets.

Regarding the Intent Recognition sub-task, accuracy is the evaluation metric of concern, following the related bibliography. Accuracy is defined as the ratio of correctly classified intents over the sum of the test examples. Nevertheless,

during our experiments, we also recorded the F1 score in the ATIS dataset, because its intents form a highly skewed distribution. However, the F1 score seemed to harmonize with the accuracy score, thus, we are finally reporting accuracy, so as to be consistent with the rest of the relevant literature. Concerning the Slot Filling sub-task, we report the micro-averaged F1 score, defined as follows:

$$\textit{micro-F1-score} = 2\frac{\textit{micro-precision} \cdot \textit{micro-recall}}{\textit{micro-precision} + \textit{micro-recall}}$$

In other words, micro-F1-score constitutes the harmonic mean of micro-precision and micro-recall which are defined as:

$$\textit{micro-precision} = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c},$$

$$\textit{micro-recall} = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c},$$

where TP_c, FP_c, FN_c represent the True Positives, False Positives and False Negatives referring to the class c respectively. In the case of BERT fine-tuning, there are two classes for the begin (B) and inside (I) tokens of each slot, respectively, plus an outside (O) class. Hence, the number of classes is equal to twice the number of slots plus one.

4.3 Training Details

Regarding the initial fine-tuning approach, we use the uncased variant of BERT-base as our datasets are already in lower case format. The maximum length of the sequences is 64. Moreover, the AdamW optimizer is used, which is a variation of the Adam optimizer using weight decay, with zero warm-up steps and a learning rate of 5e-5. The dropout rate is set to 0.1. The maximum number of epochs is 10 and 5 for the ATIS and the SNIPS datasets respectively, since SNIPS contains more training examples hence, the model converges faster. The micro F1-score of the development set is used for early stopping, so as to select the best performing model. Finally, the batch size used is 32.

For the sequence-to-sequence approach, T5-small is utilized for the experiments, since the T5-base model is quite larger and requires lots of computational resources. The maximum length of the input is set to 64 for the intent detection and also the slot filling problem where no template was used. In the case of prompting, we expand the input length to 240 in order to fit the added template phrases. AdamW

is again used as the optimizer with a learning rate of 1e-3, adhering to the value used in the original paper for fine-tuning on downstream tasks. Smaller values for the learning rate, as well as the Adafactor optimizer used in the paper were also tried out, but led to reduced performance. The maximum number of epochs is set to 8 when training on the full dataset and to 15 when simulating few-shot scenarios. The micro F1-score of the development set is again used for early stopping, whereas the batch size is set to 16.

Finally, we note that all experiments were conducted using a single Nvidia GTX-1070Ti GPU, with 8GB of VRAM.

4.4 Experimental Results

Table 4.2 depicts the performance of the fine-tuned BERT model on the ATIS and SNIPS datasets. It can be seen that this method takes full advantage of the presence of the BIO tags for slot filling while also displaying great effectiveness in the classification of the intents.

	ATIS		**SNIPS**	
	Intent Accuracy	Slot F1	Intent Accuracy	Slot F1
Attent. – BiRNN [18]	91.1	94.2	96.7	87.8
Slot-Gated [10]	94.1	**95.2**	97.0	88.8
Joint BERT	**97.0**	95.0	**98.9**	**95.8**

Tab. 4.2: Results of the fine-tuned BERT model on the ATIS and SNIPS datasets.

As far as the sequence-to-sequence approach is concerned, dealing with the intent detection problem, both models (with and without template) seem to perform on par, as Table 4.3 depicts. In the full training set case, they achieve remarkable accuracy, whereas in the few-shot scenario there is a foreseeable, relatively small performance drop. In this case, adding templates does not seem to provide clear improvements.

	Full Train Set (13K examples)	**Few-shot (100 examples)**
w/out template	97.9	91.8
with template	98.0	92.1

Tab. 4.3: Intent Detection results of T5 on the Snips dataset.

On the other hand, inspecting Table 4.4, there is a clear pattern regarding the few-shot settings in the slot filling problem. The approach that utilizes templates

substantially outperforms the method which does not. More importantly, the difference between them grows as the training set shrinks. This constitutes a strong confirmation that prompting does provide a clear edge in few-shot settings. Furthermore, when trained on the full set, T5 gains a small advantage with the usage of templates. This may imply that prompting provides a slight edge not only in few-shot scenarios, but also in full data settings of problems that are adequately complex, like the slot filling one in this case.

Finally, regarding the experiment that examines the generalizability a model gains from prompting, the results are very enlightening. The model that utilizes templates outperforms the one that does not by more than 20% in terms of micro F1-score, when evaluated on the two few-shot intents. This result provides a strong indication that prompting enhances the model's extrapolation ability to unknown intents. In a sense, the model becomes more agile and capable of transferring its "knowledge" to different domains.

As a side note, we report that while T5 was fine-tuned for the slot filling problem, slightly changing the template's content resulted in marginal performance differences, probably indicating that the model is able to adapt to small template variations, at least when the different template texts are semantically close with each other. This partly goes along with [2] which argues that fine-tuning the LM reduces the need for prompt engineering and [15] which claims that prompting is mostly robust to template choice. Nevertheless, further experiments should be conducted to reach a more reliable conclusion.

	Full Train Set	**Few-shot (500 examples)**	**Few-shot (200 examples)**	**2-Intent Few-shot (40 examples)**
w/out template	95.2	70.2	57.5	61.2
with template	**95.9**	**82.4**	**71.3**	**83.6**

Tab. 4.4: Slot Micro-F1-score for the T5-small model.

4.5 Ablation Analysis

For the traditional BERT fine-tuning approach, we perform an ablation analysis, that is we train the model for fewer epochs on the SNIPS dataset. We observe that by just training the system for 1 epoch, it outperforms the methods conducted in [18] and [10]. Specifically, it achieves 97.4% Intent accuracy and 89.2% Slot

micro F1-score, highlighting the dominance of BERT when compared to previous sequential neural network based approaches.

Regarding the sequence-to-sequence approach to the slot filling problem, we examine the influence of the encoder and the decoder parts of the network. For that reason, we successively freeze the parameters referring to each individual part thus, significantly reducing the number of trainable weights as well as the computational time. As Table 4.5 shows, the model is able to adapt to the dataset even though it trains under half of its available parameters. In particular, in the case where we keep the decoder part frozen, it displays almost on par performance with the case where we fine-tune the full model.

	Full Model Train (60m params)	**Frozen Encoder (25m params)**	**Frozen Decoder (18.8m params)**
w/out template	95.2	86.1	92.0
with template	95.9	86.0	92.6

Tab. 4.5: Slot Micro-F1 score for the T5-small model with frozen encoder and decoder parts respectively.

5 Conclusions and Future Work

In this thesis, we examined different methods for tackling the problems of Intent Recognition (IR) and Slot Filling (SF), which constitute two core tasks of Natural Language Understanding. Traditionally, rule-based and various machine learning techniques were adopted, but led to limited performance. With the recent advances in the field of Deep Learning and the advent of Pre-trained Language Models, various approaches have been proposed that address these two problems, using modern Transformer-based LMs. However, as far as the more challenging SF problem is concerned, little research has been conducted regarding the cases where there is weakly-annotated data, i.e. the so-called BIO tags are absent. In addition, in most real-world scenarios, there is limited labeled data availability, hence studying few-shot scenarios has attracted increasing attention in recent bibliography.

Considering the above problems encountered, following the line of Chen et al. [5], we initially implement a BERT model, fine-tuned to jointly model the IR and SF tasks, by exploiting the presence of BIO tags. This model significantly outperforms previous state-of-the-art approaches that used some kind of recurrent neural network structure (e.g. RNN, LSTM), on the publicly available ATIS and SNIPS datasets. These results prove that fine-tuning LMs on the SF task is the dominant approach, when BIO tags are available.

In the sequel, in order to address the problem of slot label alignment, we propose using a sequence-to-sequence framework (T5 is our model of choice). Thus, IR and SF are reformulated into a text-to-text scheme, which leads to very competitive results. Subsequently, regarding the few-shot scenario, we incorporate the paradigm of prompting by providing hand-crafted templates to the model. The model is then asked to locate the relevant information from the user utterance, by filling up these templates. In a set of experiments conducted on the SNIPS dataset, the combination of T5 with this prompting technique achieves 98% Intent accuracy and 95.9% Slot micro F1-score (after utilizing an oracle in order to know the intent in advance), when trained on the full dataset. Additionally, in low-data regimes, it substantially improves the micro F1-score metric compared to its counterpart model, which does not employ templates.

In future work, we plan to examine the concept of soft-prompts. Specifically, by slightly adapting the line of [16], we intend to replace the hand-crafted prompts with continuous vectors (embeddings of pseudo-tokens) learnt through backpropagation and either freeze the rest of the model's parameters or keep fine-tuning the model along with the embeddings of the pseudo-tokens. The reasoning behind this idea, is that providing hand-written prompts is sub-optimal and imposes constraints to the inherent knowledge or abilities of the Language Model. Hence, by letting the model decide which artificial embeddings (not necessarily corresponding to real words) best describe the corresponding slot, we optimize the prompt, which implies maximum performance.

Bibliography

[1]Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. "Augmented Natural Language for Generative Sequence Labeling". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 375–385.

[2]Robert L. Logan IV au2, Ivana Balažević, Eric Wallace, et al. *Cutting Down on Prompts and Parameters: Simple Few-Shot Learning with Language Models*. 2021. arXiv: 2106.13353 [cs.CL].

[3]Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR,* 2015.

[4]Tom Brown, Benjamin Mann, Nick Ryder, et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[5]Qian Chen, Zhu Zhuo, and Wen Wang. *BERT for Joint Intent Classification and Slot Filling*. 2019. arXiv: 1902.10909 [cs.CL].

[6]Yun-Nung Chen and Jianfeng Gao. "Open-Domain Neural Dialogue Systems". In: *Proceedings of the IJCNLP 2017, Tutorial Abstracts*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 6–10.

[7]Alice Coucke, Alaa Saade, Adrien Ball, et al. *Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces*. 2018. arXiv: 1805.10190 [cs.CL].

[8]Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. "Template-Based Named Entity Recognition Using BART". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, 2021, pp. 1835–1845.

[9]Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186.

[10]Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, et al. "Slot-Gated Modeling for Joint Slot Filling and Intent Prediction". In: *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2018.

[11]Charles T. Hemphill, John J. Godfrey, and George R. Doddington. "The ATIS Spoken Language Systems Pilot Corpus". In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. 1990.

[12]Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. "How can we know what language models know?" In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 423–438.

[13]Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2009.

[14]Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. "Leveraging Sentence-level Information with Encoder LSTM for Semantic Slot Filling". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2077–2083.

[15]Teven Le Scao and Alexander Rush. "How many data points is a prompt worth?" In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 2627–2636.

[16]Brian Lester, Rami Al-Rfou, and Noah Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 3045–3059.

[17]Xiang Lisa Li and Percy 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)*. Online: Association for Computational Linguistics, 2021, pp. 4582–4597.

[18]Bing Liu and Ian Lane. "Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling". In: *Proc. Interspeech 2016*. 2016, pp. 685–689.

[19]Yinhan Liu, Myle Ott, Naman Goyal, et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[20]Samuel Louvan and Bernardo Magnini. "Recent Neural Methods on Slot Filling and Intent Classification for Task-Oriented Dialogue Systems: A Survey". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 480–496.

[21]Colin Raffel, Noam Shazeer, Adam Roberts, et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[22]Timo Schick and Hinrich Schütze. "Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, pp. 255–269.

[23]Timo Schick and Hinrich Schütze. "It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 2339–2352.

[24]Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 4222–4235.

[25]Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[26]Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. “Bi-directional recurrent neural network with ranking loss for spoken language understanding”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 6060–6064.

[27]Zexuan Zhong, Dan Friedman, and Danqi Chen. “Factual Probing Is [MASK]: Learning vs. Learning to Recall”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 5017–5033.

List of Figures

2.1 Components of a modern Dialogue System [6]. 8
2.2 Rule-based systems [13]. 8
2.3 A Transformer consisting of 2 stacked encoders and decoders. . . . 10
2.4 BERT mask-filling pre-training objective. 11
2.5 T5 framing various NLP tasks in a text-to-text scheme [21]. 12
2.6 T5's span corruption pre-training objective [21]. 13

3.1 BERT fine-tuning for classification and NER tasks [9]. 18
3.2 Slot Filling approach using the T5 model. The model is expected to output a text for each relevant slot (with regards to the intent of the input). The keyword "none" is produced when the corresponding slot filler is not present in the utterance. 20
3.3 Slot Filling approach using the T5 model along with templates. The model is guided to produce the slot filler of concern with the utilization of templates. 21

4.1 ATIS intent distribution. 23
4.2 SNIPS intent distribution. 24

List of Tables

1.1 An example of slot annotation with the usage of BIO tags. 4
1.2 An example of slot annotation where there is no alignment between the input utterance and the slot labels. Additionally, the slot labels do not necessarily consist of an input word or span. 4

3.1 Underlying architecture comparison between BERT and T5-small. . 19

4.1 Datasets statistics. 24
4.2 Results of the fine-tuned BERT model on the ATIS and SNIPS datasets. 26
4.3 Intent Detection results of T5 on the Snips dataset. 26
4.4 Slot Micro-F1-score for the T5-small model. 27
4.5 Slot Micro-F1 score for the T5-small model with frozen encoder and decoder parts respectively. 28