FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Cross-lingual Aspect-Based Sentiment Analysis

Jakub Šmíd

PILSEN, CZECH REPUBLIC

2023

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Cross-lingual Aspect-Based Sentiment Analysis

Bc. Jakub Šmíd

Thesis advisor
Ing. Pavel Přibáň

PILSEN, CZECH REPUBLIC 2023

Citation in the bibliography/reference list:
ŠMÍD, Jakub. *Cross-lingual Aspect-Based Sentiment Analysis*. Pilsen, Czech Republic, 2023. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Ing. Pavel Přibáň.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jakub ŠMÍD**
Osobní číslo:	**A21N0073P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Mezijazyčná aspektově orientovaná analýza sentimentu**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s dostupnými daty pro aspektově orientovanou analýzu sentimentu a s aktuálními postupy jejího řešení.
2. Prozkoumejte modely založené na architektuře Transformers včetně jejich vícejazyčných verzí.
3. Na základě předchozích bodů navrhněte vhodný postup pro aspektově orientovanou analýzu sentimentu textu. Vyberte vhodný model a postup, který umožňuje provést aspektově orientovanou analýzu textu a to zároveň na více jazycích.
4. Proveďte experimenty nad vybranými daty v alespoň dvou jazycích.
5. Výsledky experimentů porovnejte a proveďte jejich kritické zhodnocení.

Rozsah diplomové práce:	**doporuč. 50 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce:	**Ing. Pavel Přibáň** Nové technologie pro informační společnost
Datum zadání diplomové práce:	**9. září 2022**
Termín odevzdání diplomové práce:	**18. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2022

Declaration

I hereby declare that this Master's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

In Pilsen, on 14 May 2023

. . . . . .

Jakub Šmíd

Acknowledgement

I would like to thank Ing. Pavel Přibáň for his exemplary supervision of my thesis, expert advice, valuable comments, willingness and patience.

Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Abstract

This thesis focuses on cross-lingual aspect-based sentiment analysis (ABSA), an understudied area in contrast to monolingual ABSA. This thesis proposes two methods that can be used with prompting and traditional-based fine-tuning. The first method is a sequence-to-sequence method that solves multiple ABSA tasks simultaneously and outperforms previous state-of-the-art results on benchmark datasets in multiple languages. Prompting improves the performance of the T5 model and its multilingual version significantly, which resulted in the best overall results among the tested models. The best cross-lingual results are also promising. The second method classifies the sentiment polarity of aspect terms and categories, establishing new state-of-the-art results in multiple languages and achieving excellent cross-lingual results, often within 2% of monolingual results. In addition, this thesis presents a newly annotated Czech dataset for ABSA.

Abstrakt

Tato práce se zaměřuje na mezijazyčnou aspektově orientovanou analýzu sentimentu (ABSA), která je na rozdíl od jednojazyčné ABSA málo probádanou oblastí. V této práci jsou navrženy dvě metody, které lze použít jak s tradičním trénováním (fine-tuning), tak s použitím techniky zvané „prompting“. První metodou je sequence-to-sequence metoda pro řešení více úloh ABSA současně a překonává předchozí state-of-the-art výsledky na referenčních datasetech. Prompting výrazně zlepšuje úspěšnost modelu T5 a jeho vícejazyčné verze, což vedlo k nejlepším celkovým výsledkům mezi testovanými modely. Slibné jsou také nejlepší mezijazyčné výsledky. Druhá metoda klasifikuje polarity sentimentu aspektových výrazů a kategorií, přičemž stanovuje nové nejlepší výsledky ve více jazycích a dosahuje vynikajících mezijazyčných výsledků, často v rozmezí 2 % od jednojazyčných výsledků. Kromě toho tato práce představuje nově anotovaný český dataset pro ABSA.

Keywords

Natural language processing • Aspect-based sentiment analysis • Machine learning • Cross-lingual aspect-based sentiment analysis • Neural networks • Prompting • Transformers • Sequence-to-sequence models

Contents

Introduction 1

With the rapid growth of social media and online reviews, sentiment analysis (SA) has become an increasingly important task in natural language processing (NLP). Aspect-based sentiment analysis (ABSA) is a subfield of sentiment analysis that aims to identify the sentiment of each aspect or feature of a product or service. ABSA has various practical applications, including product marketing, customer feedback analysis, and reputation management. However, most ABSA research has been conducted in English, and there is a lack of studies addressing the challenges of performing ABSA in other languages.

One of the main challenges of ABSA in languages other than English is the lack of annotated data. Annotated data is crucial for supervised machine learning, the most common approach for ABSA. However, manually annotating data is expensive and time-consuming, especially for languages with small speaker populations. To overcome this challenge, cross-lingual ABSA has emerged as a promising direction. Cross-lingual ABSA aims to transfer knowledge from a resource-rich language to a low-resource language, allowing the model to learn from annotated data in a source language and perform ABSA in the target language.

This thesis focuses on cross-lingual aspect-based sentiment analysis, which is challenging due to the high variability in language usage and the complexity of sentiment expressions in identifying aspect-level sentiment in customer reviews. The thesis aims to explore the state-of-the-art transfer learning methods for solving ABSA and cross-lingual ABSA, as well as to examine the datasets used in ABSA and to investigate models based on the Transformer architecture, including their multilingual versions. The primary goal of this thesis is to propose methods for solving multiple ABSA tasks, such as aspect term extraction or target-aspect-sentiment detection, based on the research of the state-of-the-art approaches for ABSA and to evaluate their effectiveness in solving these tasks in multiple languages, including cross-lingual settings. The overall objective is to contribute to the development of robust and accurate cross-lingual ABSA models that can be applied in various domains and languages.

Aspect-Based Sentiment Analysis

2

Sentiment analysis (SA) is a field of natural language processing (NLP) that aims to identify and understand subjective information in text, such as opinions, feelings, and emotions, towards a target (B. Liu, 2012). According to Mäntylä et al. (2018), sentiment analysis is a fast-growing field of study in NLP. Businesses commonly use it to determine whether their clients have a favourable, unfavourable or neutral opinion of their products or services.

Several tasks within the SA domain include document-level, sentence-level, and aspect-based-level polarity detection (Indurkhya & Damerau, 2010). However, this chapter and thesis solely focus on aspect-based sentiment analysis.

Aspect-based sentiment analysis (ABSA) is a specific type of SA that aims to identify targets described by aspect terms or aspect categories and the sentiment associated with each target, which can be sentiment polarity or opinion term (Zhang et al., 2022). Aspect terms can be expressed explicitly, such as the word *"screen"* in the sentence *"The screen is nice"*, or implicitly, as in the sentence *"It is bad"*. Aspect categories are usually predefined categories for each domain that define a unique aspect of an entity (e.g. `HARDWARE` and `SOFTWARE` for the *laptops* domain). The sentiment polarity describes the sentiment orientation over an aspect term or category (e.g. *positive* or *negative*). The opinion term expresses the sentiment towards the target (e.g. *"nice"* in the example *"The screen is nice"*). ABSA provides a more detailed analysis of customer feedback data, allowing businesses to analyze customer sentiment towards specific aspects of their products or services.

Consider the following review: *"The steak was delicious, and the service was friendly, but I did not like the beer."*. Document-level sentiment analysis would classify this sentence as *positive*. On the other hand, the aspect-based sentiment analysis would identify the specific aspect categories, aspect terms and sentiment polarities, as shown in Table 2.1, providing a more comprehensive review analysis than the document-level or sentence-level sentiment analysis. ABSA allows businesses to create better products and services that meet customers' demands.

Aspect category	Aspect term	Sentiment polarity
FOOD	steak	positive
SERVICE	service	positive
DRINKS	beer	negative

Table 2.1: Aspect categories, aspect terms and sentiment polarities for review *"The steak was delicious, and the service was friendly, but I did not like the beer."*.

2.1 Aspect-Based Sentiment Analysis Tasks

This section describes some selected tasks related to aspect-based sentiment analysis. However, it is important to keep in mind that there are additional ABSA tasks beyond the ones covered here, and different researchers or works may have their own names or definitions for these tasks.

Zhang et al. (2022) divide ABSA tasks into single and compound based on whether the desired output is a single sentiment element or multiple coupled elements. They also describe several ABSA tasks (for each, the input and output examples can be found in Table 2.2.):

- **Aspect term extraction** (ATE) focuses on extracting individual aspect terms on which the opinion is expressed in a text.
- **Aspect category detection** (ACD) aims to identify discussed aspect categories in a given text.
- **Opinion terms extraction** (OTE) is to identify opinion expressions towards an aspect. It can be divided into two tasks. The first task is aspect opinion co-extraction (AOCE), which aims to predict the aspect and opinion terms together. The second task is target-oriented opinion word extraction (TOWE), which aims to extract the corresponding opinion terms given a specific aspect term.
- **Aspect sentiment classification** (ASC) focuses on predicting the sentiment polarity for a specific aspect.
- **Aspect-opinion pair extraction** (AOPE) aims to extract the set of aspects and the set of opinions.
- **End-to-end ABSA** (E2E-ABSA) focuses on extracting the aspect term and its corresponding sentiment polarity. It has two types: joint and unified (UABSA).

- **Aspect category sentiment analysis** (ACSA) aims to detect the discussed aspect categories and their corresponding sentiment polarities.

- **Aspect sentiment triplet extraction** (ASTE) focuses on extracting all triplets (aspect term, opinion term, sentiment polarity) from a given text.

- **Aspect-category-sentiment detection** (ACSD) aims to detect all triplets (aspect term, aspect category, sentiment polarity) for a given sentence. The task is also called target-aspect-sentiment detection (TASD).

- **Aspect sentiment quad prediction** (ASQP) focuses on predicting all quadruplets (aspect term, aspect category, opinion term, sentiment polarity) from a given sentence.

Task	**Example input**	**Example output**
Single tasks		
ATE	*s*	{steak, service}
ACD	*s*	{`FOOD`, `SERVICE`}
AOCE	*s*	{steak, service}, {delicious, terrible}
TOWE	*s*, steak	delicious
	s, service	terrible
ASC	*s*, steak	*positive*
	s, service	*negative*
Compound tasks		
AOPE	*s*	(steak, delicious), (service, terrible)
E2E-ABSA	*s*	(steak, *positive*), (service, *negative*)
ACSA	*s*	(`FOOD`, *positive*), (`SERVICE`, *negative*)
ASTE	*s*	(steak, delicious, *positive*), (service, terrible, *negative*)
ACSD / TASD	*s*	(steak, `FOOD`, *positive*), (service, `SERVICE`, *negative*)
ASQP	*s*	(steak, `FOOD`, delicious, *positive*), (service, `SERVICE`, terrible, *negative*)

Table 2.2: An overview of the input and output examples for each ABSA task considering an input sentence *s*: *"The steak was delicious, but the service was terrible."*.

The rest of this section describes selected ABSA tasks for which datasets are also available.

2.1.1 SemEval-2014 Task 4

SemEval-2014 Task 4 (Pontiki et al., 2014) involves two domains (laptops and restaurants) and four subtasks: aspect term extraction (ATE), aspect sentiment classification (ASC), aspect category detection (ACD) and detecting sentiment analysis for a given aspect category.

In the ATE subtask, the objective is to identify aspect terms (e.g. *"wine"*, *"food"*, *"price"*, or *"waiter"*) in each review sentence. The aspect terms without expressed polarity towards them (neutral polarity) should also be identified.

In the ASC subtask, given the aspect terms, the task is to classify polarity (*positive*, *negative*, *neutral* or *conflict*) for each aspect term. Conflict occurs when both negative and positive sentiment is expressed for a given aspect term (e.g. *"Not the best salad in town, but it is always fresh"*).

The ACD subtask requires identifying all aspect categories from the given set (e.g. `FOOD`, `PRICE`) in each sentence. Aspect categories do not always appear as terms in sentences (e.g. aspect categories `PRICE` and `FOOD` in *"Expensive but delicious"*). No information linking an aspect with the categories of the aspect was provided.

In the last subtask, the task is to identify the polarity (same categories as in the second subtask) for the aspect category given the aspect categories for each sentence.

An English dataset is available for each domain for the first two subtasks. However, only a dataset for restaurant reviews is available for the third and fourth subtasks.

2.1.2 SemEval-2015 Task 12

SemEval-2015 Task 12 (Pontiki et al., 2015) defines an aspect category `E#A` as a combination of entity type `E` (e.g. laptop) and attribute type `A` (e.g. durability). The entity and attribute do not need to occur in a sentence. For example, in the sentence *"Delicious steak"*, the reviewer evaluates the *quality* (`A`) of the *food* (`E`) without explicitly mentioning it. The authors provided English datasets for three domains (laptops, restaurants and hotels). The task consists of two subtasks.

The first subtask is in-domain ABSA. The goal is to identify all the opinion tuples from a restaurant or laptop review. There are three types (tuple slots) of opinion tuples. The first type is the aspect category. Given the sentence with opinions, the goal is to find every entity `E` and attribute `A` pair corresponding to the opinions. The entities and attributes are predefined for each domain. An example of an entity for the restaurant domain is `RESTAURANT` and `FOOD`, of an attribute `QUALITY` and `PRICE`. The second type (required only in the restaurant domain) is opinion target

expression. The goal is to identify the linguistic expression referring to the reviewed entity `E` of each `E#A` pair in the given sentence. The slot should be "NULL" if the entity is not explicitly mentioned. The expression is defined by starting and ending position in the text. The third type is sentiment polarity. The goal is to assign a polarity label (*positive*, *negative* or *neutral*) to each given `E#A` pair. An example of an opinion tuple for a given sentence is depicted in Figure 2.1.

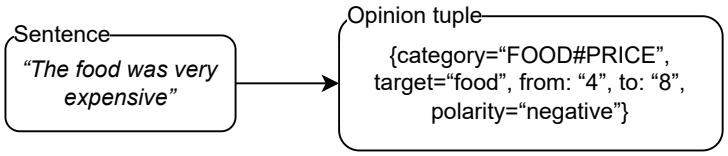


Figure 2.1: Example of an opinion tuple for a given sentence for the first subtask of the SemEval-2015 Task 12.

The second subtask is out-of-domain ABSA. The goal is to test systems in the previously unseen domain (hotel reviews) with no training data available.

2.1.3 SemEval-2016 Task 5

SemEval-2016 Task 5 (Pontiki et al., 2016) introduces three subtasks similar to those in SemEval-2015 Task 12. For this task, 19 training and 20 testing datasets are available for seven domains and eight languages. In contrast, datasets from the previous years' tasks cover fewer domains and are available only in English.

The first subtask is sentence-level ABSA, the same as the first in 2015. The third subtask is out-of-domain ABSA, similar to the second subtask in 2015, but with a changed domain (museum domain in French instead of hotel domain in English).

The second subtask is text-level ABSA. The goal is to identify a set of tuples containing category (cat) and polarity (pol) in a given customer review. These tuples summarize the opinions expressed in the review. The category is E#A tuple as in the first subtask, and polarity is either positive, neutral or negative. Figure 2.2 shows an example of opinion tuples for a given review containing two sentences.

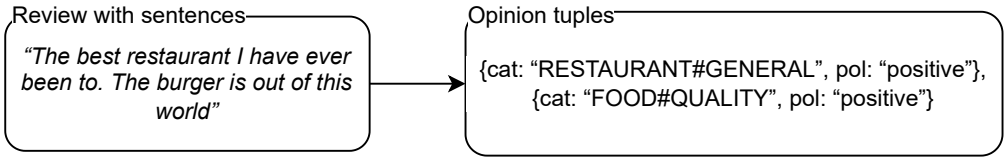


Figure 2.2: Example of an opinion tuple for a given sentence for the second subtask of the SemEval-2016 Task 5.

2.1.4 SentiHood

Saeidi et al. (2016) introduce the SentiHood dataset for targeted aspect-based sentiment analysis (TABSA). The dataset is derived from a question-answering platform where users discuss urban neighbourhoods. Previously mentioned tasks can only handle cases where all opinions about the target entity (e.g. restaurant or hotel) refer to the same entity (e.g. a single restaurant). On the other hand, the TABSA task also addresses cases where, for instance, more than one restaurant is discussed, and restaurants for which opinions are expressed are explicitly mentioned. Consider the following example (where aspects are underlined and specific entities are highlighted in bold): *"The space design is good in **Boqueria**. Still, the service is horrid. On the other hand, the staff in **Gremio** is amiable, and the food is always delicious."*. Tasks described earlier can only recognize positive and negative opinions expressed about the aspect "service". The TABSA task can also identify the specific target entity for these opinions (in this case, Gremio and Boqueria).

2.1.5 SemEval-2022 Task 10

SemEval-2022 Task 10 (Barnes et al., 2022) deals with structured sentiment analysis. The task is to find all opinion tuples $O_i, \ldots, O_n$ in a text. Each opinion tuple O_i is a tuple (h, t, e, p) that implicitly defines the connections between the components of a structured sentiment graph by having a holder h express a polarity p towards a target t via a sentiment expression e. Holders and targets can be null. Figure 2.3 shows an example of a structured sentiment graph.

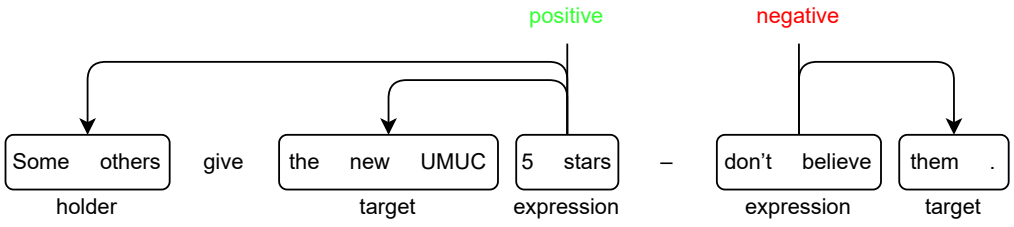


Figure 2.3: A structured sentiment graph composed of a holder, target, sentiment expression, their relationship and a polarity attribute (adapted from Barnes et al., 2022).

Seven datasets in five languages (English, Norwegian, Basque, Catalan and Spanish) are available for this task. The task consists of two subtasks. One subtask is monolingual, where models are trained and tested on data in the same language. The second subtask is cross-lingual, where the Catalan, Basque and Spanish datasets are used for testing, and any other language can be used for training. The second subtask explores the ability of models to generalize across languages.

2.2 Focus of the Thesis

The primary objective of this thesis is to address subtask 1 of the SemEval 2016 Task 5, which involves identifying aspect categories, aspect terms, (aspect category, aspect term) tuples, and sentiment polarity for given (aspect category, aspect term) tuples. Additionally, the thesis deals with the target-aspect-sentiment detection task. The experiments are conducted on datasets for the restaurant domain in all available languages and include monolingual and cross-lingual approaches.

Transformers 3

This chapter briefly introduces the pre-trained models for language representation, then describes sequence-to-sequence models, the Transformer architecture, tokenization and a few chosen models based on the Transformer architecture.

3.1 Pre-trained Models

Han et al. (2021) discuss the challenges associated with training deep neural networks such as CNN or RNN, which often suffer from overfitting due to limited training data and a large number of parameters. Efforts have emerged to address this issue by manually creating high-quality datasets for training neural networks for specific tasks (Deng et al., 2009; Lin et al., 2014). However, this is a time-consuming and expensive process. **Transfer learning** (Pan & Yang, 2009; Thrun & Pratt, 1998) offers an alternative approach by using previously acquired knowledge to solve new problems instead of training a model from scratch.

Transfer learning involves pre-training a model on one or more source tasks to capture information and knowledge and then fine-tuning it for target tasks to transfer the captured knowledge. This technique can help models perform well on target tasks even with limited samples, thanks to the richness of knowledge gained during the pre-training phase.

Two widely researched pre-training techniques in transfer learning are feature and parameter transfer (Han et al., 2021). Parameter transfer involves sharing model parameters between source and target tasks to transfer knowledge (Evgeniou & Pontil, 2004). These methods transfer the knowledge that they first pre-encode into the shared model parameters by fine-tuning the pre-trained parameters using the data from the target tasks. Feature transfer pre-trains effective feature representations to pre-encode knowledge across tasks and domains, significantly enhancing model performance on target tasks (Argyriou et al., 2006).

Another approach to deal with a limited amount of labelled data is to use unsupervised and self-supervised learning methods. Both methods use unlabelled data, of which a large amount is available. Self-supervised training uses unlabelled data

with the input data alone as supervision to extract knowledge. On the other hand, unsupervised learning focuses on detecting patterns in data, such as clustering.

Figure 3.1 shows the spectrum of pre-training methods. More details can be found in (Han et al., 2021).

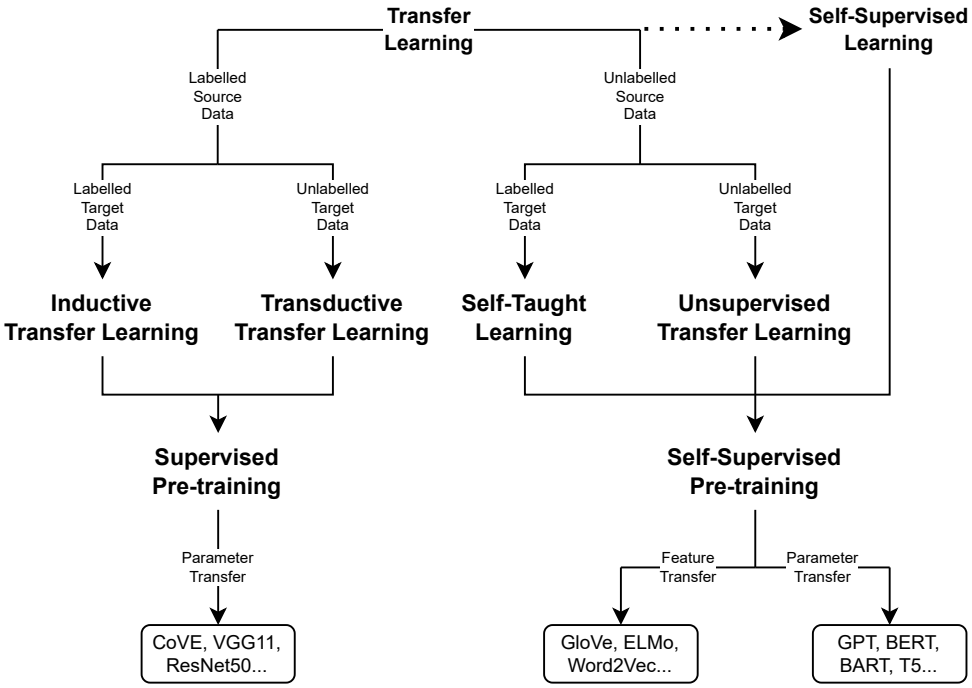


Figure 3.1: The spectrum of pre-training methods (adapted from Han et al., 2021).

Word2Vec (Mikolov et al., 2013) and *GloVe* (Pennington et al., 2014) are examples of pre-trained networks that capture the semantic meanings of words and are used to create embeddings for different words. However, these models have a disadvantage in that they cannot distinguish between *polysemous words*, which are words with multiple meanings. The reason is that these models always provide the same vector for the same word regardless of its meaning given by the surrounding words. For example, the word *"crane"* occurring in the context of *"that bird is a crane"* and *"they had to use a crane to lift the object"* always has a different meaning but is expressed by a single static vector. To mitigate this problem, pre-trained recurrent neural networks, such as **ELMo** (from *Embeddings from Language Models*) introduced by Peters et al. (2018), provide contextualized word embeddings. However, the size of these models still limits their performance.

The pre-trained models based on the Transformer architecture (described later in this chapter) perform well in language generation and understanding (Han et al., 2021). These models can deal with polysemous words and capture factual knowledge and semantic and lexical structures from texts. As the size of the model grows, so

does its performance. Fine-tuning these models yields excellent performance in downstream NLP tasks (Han et al., 2021).

3.2 Sequence-to-Sequence Model

To understand transformer design, it is initially necessary to introduce the concept of **sequence-to-sequence** (Seq2Seq) models. These models map sequences of input vectors to sequences of output vectors (Sutskever et al., 2014). Sequence-to-sequence models are commonly used in NLP tasks like machine translation, question answering, summarization and code generation.

Sequence-to-sequence models consist of an **encoder** and a **decoder** and, therefore, are also referred to as **encoder-decoder** models. The encoder maps the input sequence to a *context vector*, a function of the hidden contextualized representation of the input. The decoder then generates an output sequence based on this context vector. Depending on the task, the output sequence can be words (in the same or another language), symbols or a copy of the input sequence. Figure 3.2 illustrates the encoder-decoder architecture.

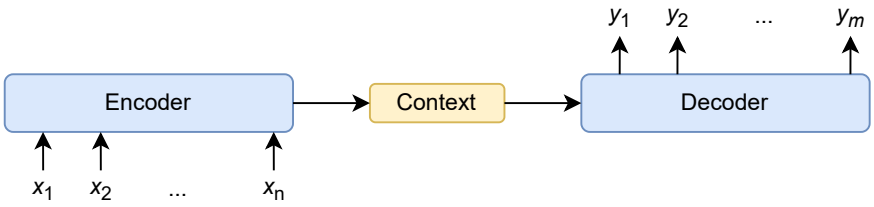


Figure 3.2: The encoder-decoder architecture.

Sequence-to-sequence models are *conditional language models* because the decoder predicts the next word y_t of the target sequence y of length T based on the previous words in that sequence and is also conditioned on the input sequence x. Sequence-to-sequence models calculate the conditional probability $P(y|x)$ as

$$\begin{aligned} P(y|x) &= P(y_1|x)P(y_2|y_1,x)\dots P(y_T|y_1,\dots,y_{T-1},x) \\ &= \prod_{t=1}^{T} P(y_t|y_1,\dots,y_{t-1},x), \end{aligned} \tag{3.1}$$

where $P(y_t|y_1,\dots,y_{t-1},x)$ is the probability of next target word y_t given previous target words and input sentence x (Jurafsky & Martin, 2009).

The decoder and encoder can be implemented using various sequence architectures, including *recurrent neural networks* (RNN) (Elman, 1990), *gated recurrent unit* (GRU) (Cho et al., 2014), *long short-term memory* (LSTM) (Hochreiter & Schmidhuber, 1997), *convolutional neural networks* (CNN) (Le Cun et al., 1989) and *Transformers*.

3.2.1 Decoding Algorithms

As described before, sequence-to-sequence models estimate the probability of word y_t in a sequence given the previous words $y_1, \ldots, y_{t-1}$ and input sentence x, which can be denoted as $P(y_t|y_1, \ldots, y_{t-1}, x)$. The probability distribution can be derived by calculating the probability for each word in the vocabulary. Based on this distribution, it can then be decided which word will be generated next. The effective choice of the following word to be generated depends significantly on the chosen method of using the probability distribution. This aspect significantly impacts the overall quality of the generated text. This subsection gives a brief overview of text-generating decoding algorithms.

3.2.1.1 Greedy Search Decoding

The first and simplest decoding algorithm is **greedy search**. Greedy search selects the word y_t with the highest probability on each step t as

$$y_t = \underset{w \in V}{\operatorname{argmax}} P(w|y_1, \ldots, y_{t-1}, x). \tag{3.2}$$

However, this algorithm has several problems (Jurafsky & Martin, 2009), including the inability to undo an earlier decision, which can lead to incorrect predictions. Figure 3.3 shows an example of greedy search decoding.

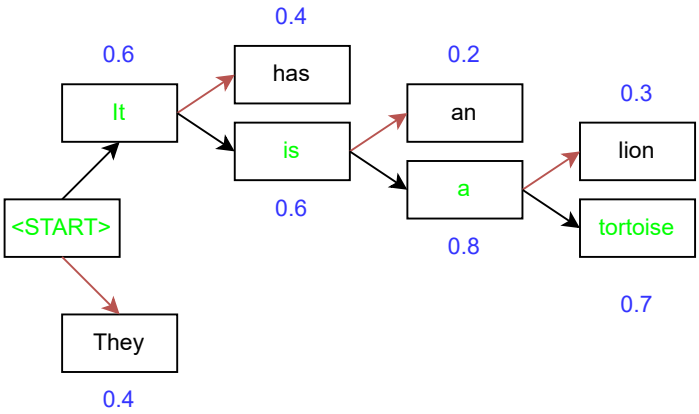


Figure 3.3: Visualization of the greedy search decoding.

3.2.1.2 Beam Search Decoding

Beam search (Freitag & Al-Onaizan, 2017) is an algorithm used to generate text by keeping track of the k most probable partial output sequences (*hypothesis*) on

each step. The k is called *beam size* or *width* and is typically set between 5 and 10 to balance computation cost and performance. When k is set to 1, the algorithm becomes equivalent to greedy search decoding. The partial sequence of length t is composed of words $y_i, \ldots, y_t$. A hypothesis $y_1, \ldots, y_t$ has a score defined as

$$\text{score}(y_1, \ldots, y_t) = \sum_{i}^{t} \log P(y_i | y_1, \ldots, y_{i-1}, x). \tag{3.3}$$

The scores are all negative; the higher the score, the better. At each step, the beam search finds the top k next words for each of the k hypotheses and calculates scores. Only the k hypotheses with the highest scores of these k^2 hypotheses are kept. Hypotheses can have different lengths, depending on the step when beam search decoding produced the token indicating sentence end. However, longer hypotheses tend to have lower scores than shorter ones, so Equation 3.3 is normalized by dividing it by the length of the hypothesis as

$$\text{score}(y_1, \ldots, y_t) = \frac{1}{t} \sum_{i}^{t} \log P(y_i | y_1, \ldots, y_{i-1}, x). \tag{3.4}$$

The beam search algorithm is not guaranteed to find the optimal solution, but it is a popular and effective decoding method (Jurafsky & Martin, 2009). Figure 3.4 shows an example of the beam search decoding with $k = 2$.

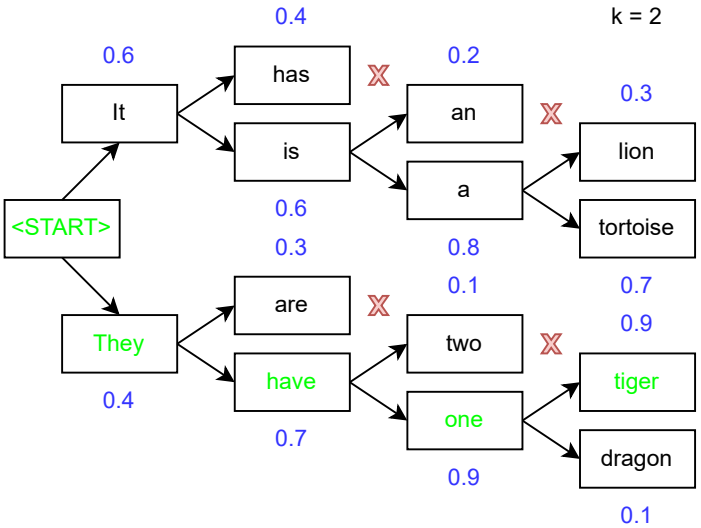


Figure 3.4: Visualization of the beam search decoding with $k = 2$.

3.2.1.3 Sampling

The decoding algorithms mentioned earlier can be altered by incorporating random **sampling**. This technique randomly selects the next word according to its conditional probability distribution. An example is *Top-K sampling*, where the algorithm chooses only the top k words. The benefit of sampling is that it introduces randomness into the decoding process, which can lead to more diverse output sequences. However, the random nature of word selection may result in meaningless terms being generated in certain contexts.

3.3 Transformer Architecture

The **Transformer** architecture introduced by Vaswani et al. (2017) is a variation of the encoder-decoder models built on the **attention** mechanism.

3.3.1 Attention

The attention mechanism enables the decoder to identify which parts of the input sequence are relevant at each decoding step. In the Transformer, the attention mechanism is implemented using self-attention and cross-attention.

Self-attention in Transformers (Vaswani et al., 2017), also referred to as *scaled dot-product attention*, is used to compute the importance of each word in a sequence of length N. It is computed as

$$\text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathsf{T}}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3.5}$$

where $\mathbf{Q} \in \mathbb{R}^{N\times d_k}$, $\mathbf{K} \in \mathbb{R}^{N\times d_k}$ and $\mathbf{V} \in \mathbb{R}^{N\times d_v}$ are the **query**, **key** and **value** matrices, respectively, and d_k is the dimensionality of the query and key vectors. The keys, queries and values are vectors created by multiplying the input word vectors (**embeddings**) by three learned weight matrices. The dot product of the $\mathbf{Q}$ and $\mathbf{K}$ matrix determines how each word influences all other words in the sequence. The result is then scaled by the keys' dimension and normalized using a *softmax* function. The values are then weighted according to the softmax scores and summed up to produce the final attention output. When used in decoders, self-attention is masked to prevent the decoder from accessing future words. An example of the self-attention mechanism with generated attention scores is illustrated in Figure 3.5.

The decoder part of the Transformers also uses a **cross-attention**. The computation of cross-attention is the same as self-attention. The difference is that the input of self-attention is a single embedding sequence. The cross-attention, on the other hand, combines two separate sequences. The queries come from the previous layers of the decoder, while the keys and values come from the encoder.

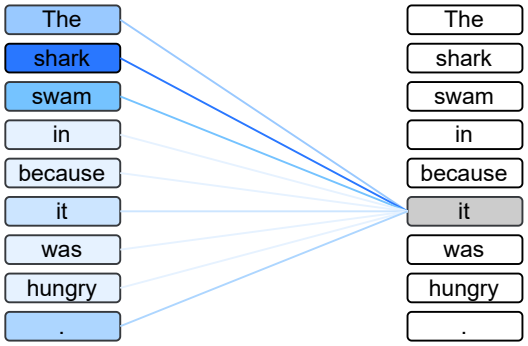


Figure 3.5: An illustration of the self-attention mechanism of the Transformer when encoding the word *"it"*, where the darker the colour of the square, the larger the corresponding attention score.

Transformers use **multi-head attention** layers to capture different types of relationships between inputs. Individual words in a sentence can be related to each other in many different ways simultaneously. For instance, various syntactic and semantic relations exist between verbs and their arguments in a sentence. Multi-head attention layers consist of multiple attention layers, called **attention heads**, and project the keys, values and queries multiple times with different linear layers. Outputs of all the attention heads are combined and projected by another linear layer. Figure 3.6 shows the multi-head attention mechanism.

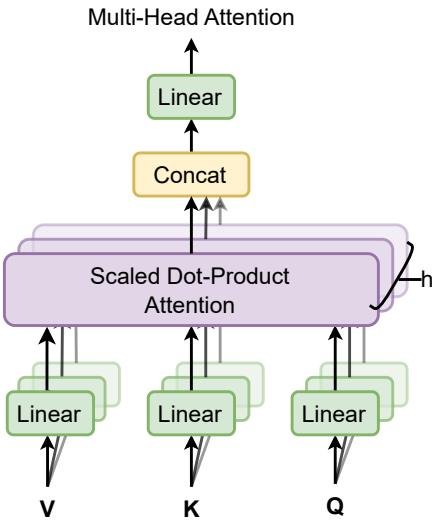


Figure 3.6: Multi-head attention consists of multiple attention layers running in parallel (adapted from Vaswani et al., 2017).

3.3.2 Internal Structure of the Transformer

Figure 3.7 shows the Transformer architecture. First, an **embedding layer** converts the input and output tokens to vectors (**embeddings**). Then, positional encoding is applied to preserve positional information. The encoder and decoder consist of multiple blocks with **multi-head attention** layers and **position-wise feed-forward networks**. As mentioned earlier, the decoder has an additional layer that performs multi-head attention over the encoder output.

The Transformer uses residual connections followed by layer normalization around each sub-layer. Residual connections (He et al., 2016) improve the model's training, while layer normalization (Ba et al., 2016) helps the model train faster. During input processing, the model can access all previous inputs but not future ones. One of the advantages of Transformers is that the training can be parallelized because each word's computation is performed independently of all other computations.

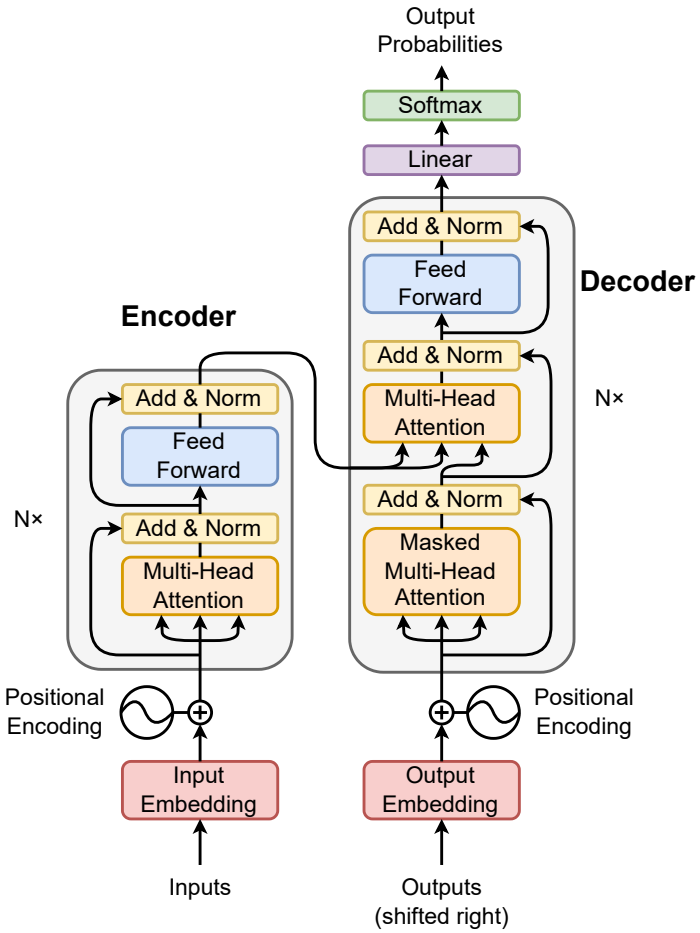


Figure 3.7: Architecture of the Transformer (adapted from Vaswani et al., 2017).

3.4 Tokenization

The NLP models often require some form of text preprocessing, and **tokenization** is one of the most common methods. Tokenization involves splitting words into tokens (as shown in Figure 3.8), which are then stored in a vocabulary and represented as indices. The tokenization process may also remove specific characters from the text, such as punctuation (Manning et al., 2008). There are several factors to consider when choosing a tokenization method.

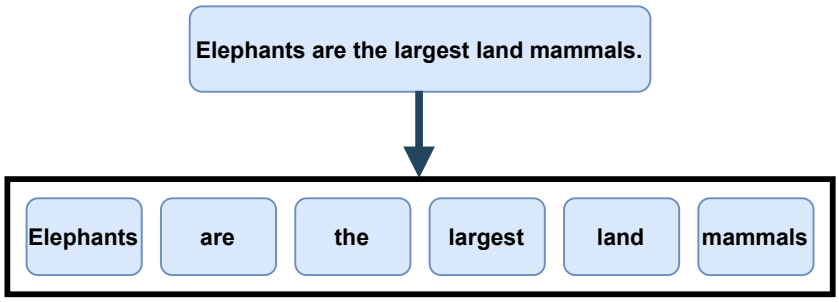


Figure 3.8: An example of splitting a sentence into individual tokens (tokenization).

The first consideration is the vocabulary size. For instance, a simple tokenization method that splits the input text by whitespace lead to an extensive vocabulary. The vocabulary size can be reduced by choosing a minimal frequency of occurrence of the tokens in the corpus, e.g. 5. However, even with this reduction, the vocabulary size is often too big. For example, the pre-trained *Word2Vec* (Mikolov et al., 2013) embeddings have a vocabulary size of 3 million.

The second consideration is *out-of-vocabulary* (OOV) tokens, which are not part of the vocabulary. Examples of OOV tokens include word variations (e.g. *taaaasty*), misspellings (e.g. *laern*), and novel items (e.g. *Transformerify*). One approach is to ignore these tokens. The second possible solution is to map them to a single predefined unique token <UNK>. The finite vocabulary assumption is more challenging for languages with more complex *morphology* (the structure of words), as longer and more complex words are less likely to be included in the vocabulary because they occur less frequently. For instance, Czech words can have many conjugations (prefixes, suffixes). Each conjugation encodes essential information about the sentence that more words in English might represent. For example, *plavu – neplavou* in Czech represents *I swim – They do not swim* in English.

This section further describes tokenization methods that can reduce vocabulary size and solve the OOV problem. The methods, called **subwords models**, look at the internal structure of words and learn a vocabulary of parts of words (subword tokens). Figure 3.9 shows an example of subword tokenization.

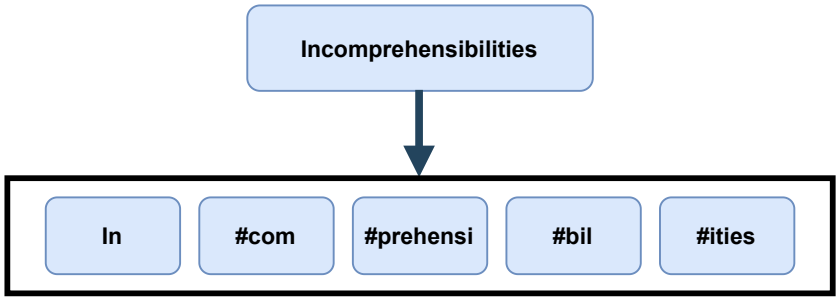


Figure 3.9: Example of subword tokenization of the word *Incomprehensibilities*.

3.4.1 Byte-Pair Encoding

Byte-pair encoding (BPE) is a simple and practical strategy for defining a subword vocabulary (Jurafsky & Martin, 2009). The algorithm starts with a vocabulary containing all individual characters. Next, it finds the most common adjacent characters in a corpus text, merges these two characters, adds the newly created subword to the vocabulary and replaces every instance of the character pair in the corpus with the new subword. This process is repeated until the desired vocabulary size k is reached. The input for the algorithm is usually pre-tokenized text (e.g. on white spaces), so it does not merge across word boundaries.

3.4.2 WordPiece

WordPiece (Schuster & Nakajima, 2012) is a similar algorithm to BPE. Like BPE, it starts by initializing the vocabulary with all the characters. However, it differs in the merging strategy. Instead of choosing the most frequent character pair, WordPiece chooses the one that maximizes the likelihood of the training corpus once added to the vocabulary.

3.4.3 Unigram

The **unigram** algorithm (Kudo, 2018) starts with a vast vocabulary and iteratively reduces it. The algorithm defines a loss over the training corpus of words $x_1, \ldots, x_N$ given a unigram language model and current vocabulary as

$$\mathcal{L} = -\sum_{i=1}^{N} \log\left(\sum_{x \in S(x_i)} p(x)\right), \tag{3.6}$$

where $S(x_i)$ is a set of all possible tokenizations for a word x_i. At each step, the algorithm removes p (usually 10 or 20) per cent of symbols in the vocabulary. The symbols to be removed are selected so that their removal results in the smallest loss increase.

3.4.4 SentencePiece

All of the algorithms described so far require pre-tokenized input text. That can lead to problems when dealing with languages that do not separate words by spaces (e.g. Chinese and Japanese). **SentencePiece** (Kudo & Richardson, 2018) mitigates this problem by using the raw input directly. To construct the vocabulary, it then uses BPE or the unigram algorithm. The algorithm includes the white space in the initial vocabulary. Therefore, it can represent multi-word expressions like *"Los Angeles"* with a single token.

3.5 Models Based on the Transformer Architecture

This section describes some selected models based on the Transformer architecture that are important in the NLP field or related to this thesis.

3.5.1 GPT

The **GPT** model (from *Generative Pre-trained Transformer*) introduced by Radford et al. (2018) is a neural network model based on the Transformer decoder. The GPT model utilizes autoregressive language modelling to predict the next word in a sequence by considering the previous words, as depicted in Figure 3.10.

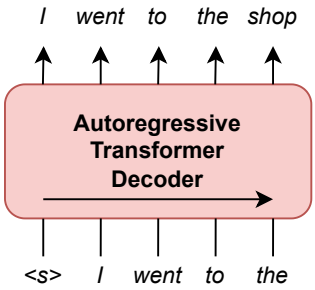


Figure 3.10: Example of autoregressive generation of tokens in GPT. Only left-side context is used to generate the next word.

The first version of GPT (GPT-1) has 12 layers and 117 million parameters and uses byte-pair encoding with 40,000 merges. The model is first pre-trained on a large unlabelled corpus to learn general language representations. Afterwards, the model is fine-tuned on specific downstream tasks, such as sentiment analysis or classification, where the model can take advantage of previously acquired knowledge from the pre-training.

A GPT-2 model (Radford et al., 2019) is an improved model with more parameters, specifically 1.5 billion. It has a decoder with 48 layers. The authors show that this large model is efficient for *zero-shot learning*, where no training examples are provided (therefore, there is no fine-tuning).

A GPT-3 model (Brown et al., 2020) has even more parameters than the previous (175 billion) and 96 decoder layers. The authors further show that the model is excellent for *zero-shot* and *few-shot* learning. During the few-shot learning, only a few examples are provided for fine-tuning. All GPT models have achieved new state-of-the-art results on various tasks at the time of their introduction.

3.5.2 BERT

The **BERT** model (from *Bidirectional Encoder Representations from Transformers*) introduced by Devlin et al. (2019) is based on the Transformer encoder, unlike the GPT model, which uses the decoder stack. The improvement over the original encoder model is the bidirectional training, which allows left-to-right and right-to-left processing of the sequence. Bidirectional processing is particularly useful for named entity recognition, where information from the right context can be helpful. Devlin et al. (2019) propose two objectives for model pre-training: masked language modelling and next sentence prediction.

During **masked language modelling** (MLM), 15% of (sub)word tokens are randomly selected. Of these, 80% are replaced with a `[MASK]` token, 10% with a random token from vocabulary, and 10% are left unchanged. The model then tries to predict the correct token. Figure 3.11 shows an example of masked language modelling.

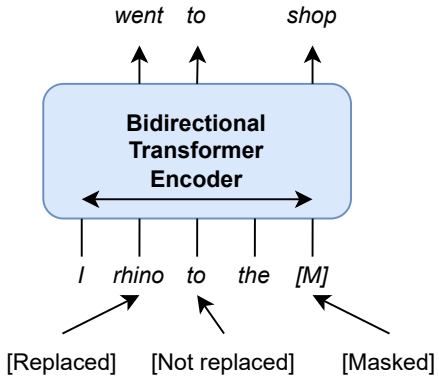


Figure 3.11: Example of masked language modelling with three lost terms for the sentence. The model uses both left-side and right-side context.

Next sentence prediction (NSP) is a task where the model tries to determine if

two sentences are in the correct order. In 50% of the instances, the inputs are pairs in which the second sentence is the next one in the original text, and in the remaining 50%, the second sentence is randomly selected from the corpus. This way, the model should learn and understand the relationships between two different pieces of text.

Initially, two BERT models were released. The first was called BERT-base, with 12 layers and 110 million parameters; the second was BERT-large, with 24 layers and 340 million parameters. Fine-tuning BERT led to new state-of-the-art results on various tasks (Devlin et al., 2019).

3.5.3 RoBERTa

The **RoBERTa** model (from *A Robustly Optimized BERT Pre-training Approach*) presented by Y. Liu et al. (2019) is based on the BERT model but has some modifications.

The first change is that the RoBERTa model uses dynamic masking instead of static masking in the BERT model. In dynamic masking, different parts of the sentence are masked for different pre-training epochs, making the model more robust. In static masking, the same part of the sentence is masked in each epoch.

The second change is that the RoBERTa model does not use the NSP task, which was observed to be not very useful. Therefore, the RoBERTa model uses only the MLM task.

The third change is the size of the data used for pre-training the model. The BERT model is pre-trained on the *BooksCorpus* (Zhu et al., 2015) (800 million words) and English Wikipedia (2.5 billion words), with 16 GB of text. In addition to these datasets, the RoBERTa model is also pre-trained on other datasets, specifically *OpenWebText*, *CC-News* (CommonCrawl News), and *Stories* (Trinh & Le, 2018). The total size of these datasets is around 160 GB.

Another change is the size of the batch used. The RoBERTa model uses a batch size of 8,000 with 31,000 steps. For comparison, the BERT model uses a batch size of 256 with 1 million steps. This setting increases the performance of the RoBERTa model (Y. Liu et al., 2019).

Furthermore, the RoBERTa model also uses a larger BPE vocabulary size of 50,000 sub-word units compared to only 30,000 used in the BERT model, resulting in a larger number of parameters for the RoBERTa model compared to the BERT model.

The RoBERTa model has two versions: RoBERTa base with 12 layers and 125 million parameters and RoBERTa-large with 24 layers and 355 million parameters. The RoBERTa model outperforms the BERT model in many tasks (Y. Liu et al., 2019).

3.5.4 ELECTRA

Clark et al. (2020) introduce the **ELECTRA** model (from *Efficiently Learning an Encoder that Classifies Token Replacements Accurately*), another variant of the BERT model that is lighter than its predecessor and uses a generator-discriminator structure.

The ELECTRA model uses **replaced token detection** (RTD) instead of masked language modelling. In replaced token detection, instead of masking the token, the token is replaced by an incorrect token. The model learns to classify whether each token is original or a replacement. This approach trains two neural networks, a generator and a discriminator, both based on the encoder stack of the Transformer. The generator is trained to perform MLM and maximize the likelihood of the randomly masked-out tokens. The discriminator is trained to distinguish between original and generator-replaced tokens in the data. Figure 3.12 shows the overview of the RTD pre-training task.

The discriminative task has the advantage of being more computationally efficient since the model learns from all input tokens and not just a small subset of masked tokens. This modification leads to faster training and higher accuracy on downstream tasks than the BERT model when fully trained (Clark et al., 2020). The authors also improved the pre-training efficiency by sharing weights between the discriminator and the generator.

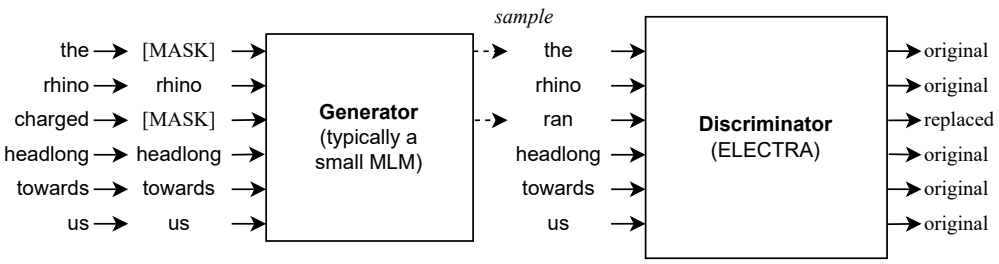


Figure 3.12: An overview of replaced token detection proposed by Clark et al. (2020).

3.5.5 BART

The **BART** model (Lewis et al., 2020) is a denoising sequence-to-sequence model based on the original Transformer architecture. It improves the learning process by combining the strengths of *Bidirectional* and *Auto-Regressive Transformers* (hence the name BART). Similar to BERT, BART uses a bidirectional encoder over corrupted text. The decoder then generates a sequence of words in an autoregressive manner, as in GPT, to reconstruct the original uncorrupted sequence.

The authors create a dataset of corrupted text using different transformations (shown in Figure 3.13), which can be combined arbitrarily:

- **Token masking** replaces random tokens with `[MASK]` tokens (same as in BERT).
- **Token deletion** deletes random tokens.
- **Text infilling** adds an extra `[MASK]` token to a random position or replaces a span of tokens with a `[MASK]` token. The span length is sampled from a *Poisson* distribution with $\lambda = 3$.
- **Sentence permutation** shuffles sentences (given the full stops) in random order.
- **Document rotation** rotates the document around a randomly selected token so it begins with the selected token.

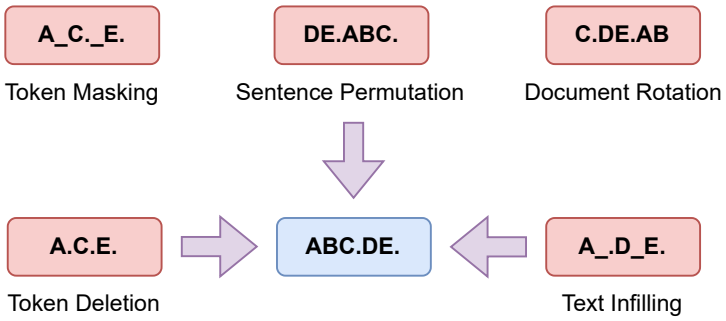


Figure 3.13: Transformations used in the BART model to noise the input (adapted from Lewis et al., 2020).

3.5.6 T5

The **T5** model (from *Text-To-Text Transfer Transformer*) proposed by Raffel et al. (2020) is a sequence-to-sequence model, similar to BART, that takes text as input and produces text as output. During pre-training, the T5 model replaces arbitrary spans of tokens in the input with unique sentinel tokens. The task is to decode the masked spans. The output sequence consists of the dropped spans of tokens separated by the sentinel tokens used in the input, plus the final sentinel token that marks the end of the target sequence, as shown in Figure 3.14. Unlike BART, T5 only predicts tokens replaced by the mask token during pre-training, which differs from BART's approach that reconstructs the entire sentence.

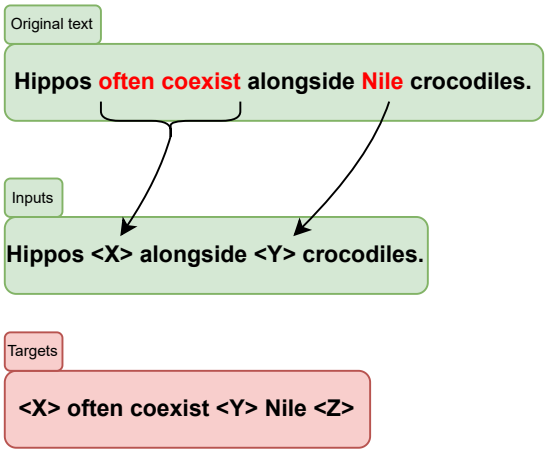


Figure 3.14: The objective used in the T5 model. The red words represent the tokens that are randomly chosen for corruption. Unique sentinel tokens replace each consecutive span of corrupted tokens. The target sequence contains the dropped spans of tokens separated by the sentinel tokens used in the input, plus the last sentinel token that marks the end of the target sequence.

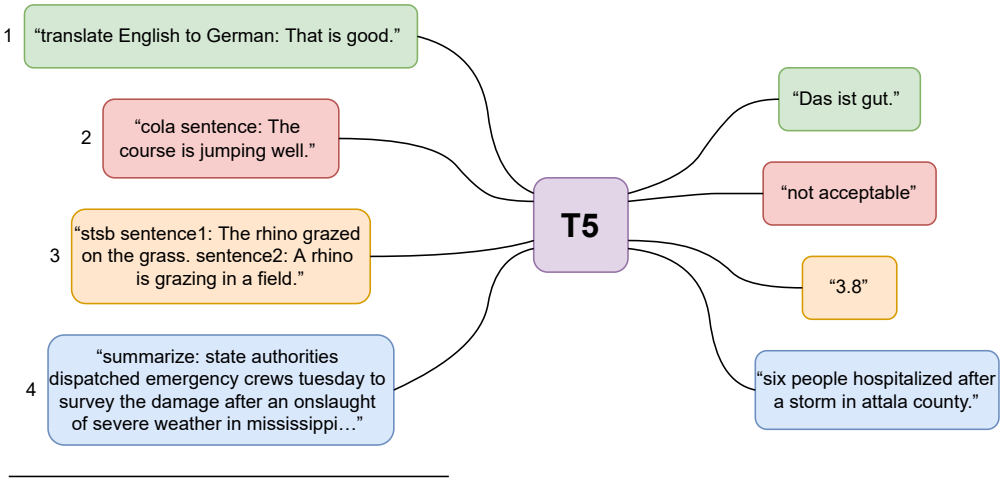
Raffel et al. (2020) also fine-tuned the model on various downstream tasks, including text classification, question answering, summarization and machine translation. The authors also introduce a concept of task-specific prefixes to convert every considered task into a text-to-text format. By adding the prefix to the original input, the model chooses which task to perform, as shown in Figure 3.15.

The authors also present a new dataset, *Colossal Clean Crawled Corpus* (C4), which contains about 750 GB of English text. They use the corpus to pre-train the T5 model and experiment with their denoising objective.

T5v1.1 is an improved version of the original T5 model that employs the GEGLU activation function (Shazeer, 2020) instead of ReLU (Glorot et al., 2011). It is pre-trained only on the C4 dataset, excluding any supervised training. Therefore it is not advantageous to use a task prefix during single-task fine-tuning. **ByT5** (Xue et al., 2022) is another T5 variant that operates with sequences of characters instead of tokens corresponding to words or subword units.

3.5.7 Other Languages

All of the models mentioned so far are pre-trained on English data. However, there are models pre-trained on data from other languages. For instance, **RobeCzech** (Straka et al., 2021) and **Czert** (Sido et al., 2021) are models pre-trained on Czech data based on the RoBERTa and BERT models, respectively.



[1] machine translation task

[2] The Corpus of Linguistic Acceptability

[3] Semantic Textual Similarity Benchmark

[4] summarization task

Figure 3.15: A diagram of the text-to-text framework introduced by Raffel et al. (2020).

3.5.8 Multilingual Models

In addition, several Transformer-based models have multilingual variants pre-trained on data from multiple languages. Examples of these models are mBERT (Devlin et al., 2019), XLM (Conneau & Lample, 2019), XLM-RoBERTa (Conneau et al., 2020), mT5 (Xue et al., 2021) and mBART (Y. Liu et al., 2020; Tang et al., 2020).

Prompt-Based Fine-Tuning

4

This chapter provides an overview of *prompt-based fine-tuning* (*prompting*), a relatively new paradigm in NLP (P. Liu et al., 2023), and compares it to traditional supervised learning.

4.1 Traditional Supervised Learning

In traditional supervised learning, a model is trained to predict the output y from the input x as $P(y|x;\Theta)$. The output y can be, for instance, a label or text. Input-output pairs are required to learn the model parameters Θ. Figure 4.1 shows an example of inputs and outputs for text classification and machine translation in traditional supervised learning.

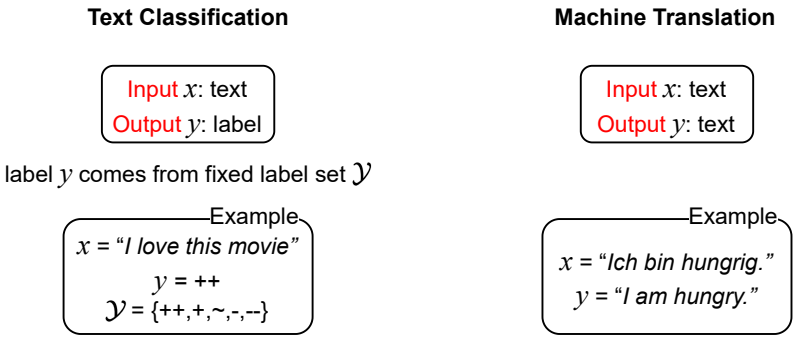


Figure 4.1: Examples of inputs and outputs for text classification and machine translation in traditional supervised learning.

Traditional supervised learning has some disadvantages. The first disadvantage is that a large amount of labelled data is required for training to achieve satisfying results. However, retrieving large amounts of labelled data is time-consuming, expensive and often difficult. The few-show learning is a way to mitigate this problem, but the results are not always satisfying compared to training on large amounts of

data. The second problem is that introducing new parameters to an existing model is often necessary. For example, a binary classification task with a BERT-large model requires additional 1,024 × 2 parameters. The third problem is that the objectives differ between the pre-training stage, which often uses masked language modelling and similar techniques, and the downstream tasks. Prompting tries to eliminate these problems.

4.2 Prompting

Prompting is a technique that encourages a pre-trained model to make specific predictions by providing a prompt specifying the task to be done. Prompt-based learning is based on language models that model the probability $P(x; \Theta)$ of text x. To modify the initial input x into a *prompt* x', prompting uses a *template* with two slots: an input slot `[X]` for input x and an answer slot `[Z]` for answer z, then mapped to output y. An example of a template for machine translation is "Czech: `[X] English: [Z]`". Figure 4.2 compares the traditional and prompting formulation for text classification.

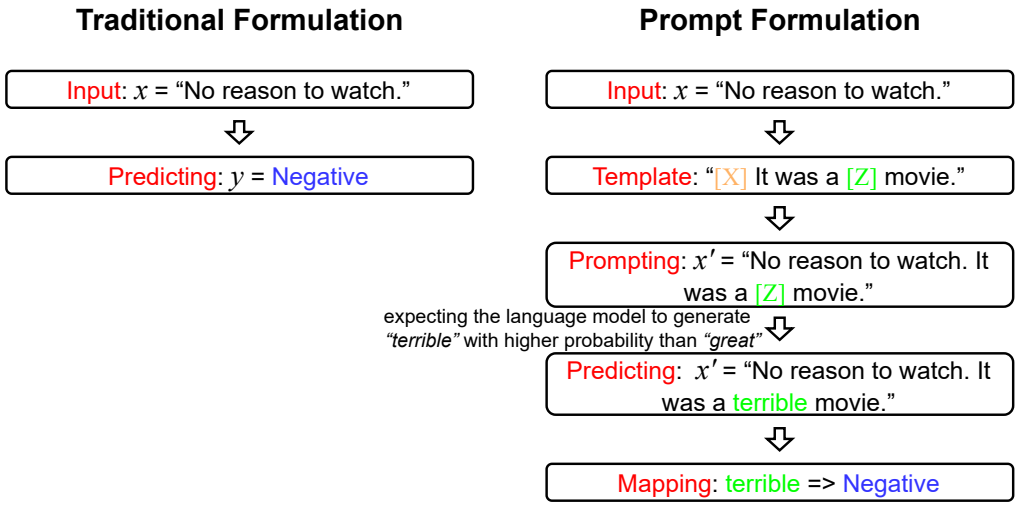


Figure 4.2: Comparison of traditional and prompting formulation for text classification.

The prompt has unfilled slots, meaning the original task can be formulated as a (masked) language modelling problem. The language model is then used to probabilistically fill the blank slots to produce a final string $\hat{x}$, from which the final output y can be inferred. Figure 4.3 compares pre-training, fine-tuning and prompting for a sentiment classification task with added demonstrations (*answered prompts*) in prompt-based fine-tuning, which can help guide the model to correct predictions, as shown by Gao et al. (2021).

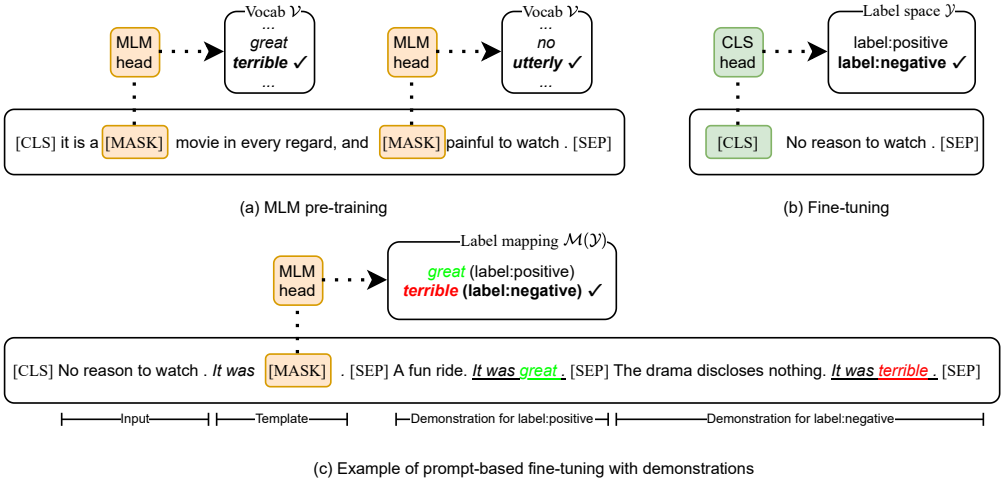


Figure 4.3: An illustration for (a) masked language model (MLM) pre-training, (b) traditional fine-tuning and (c) prompt-based fine-tuning with demonstrations for a sentiment classification task. Underlined text is the task-specific template, and coloured words are answers for the task (in this case, label words). Demonstrations for each label can be added to lead the model further to correct predictions (adapted from Gao et al., 2021).

4.2.1 Answer Search and Mapping

To create a *filled prompt* by filling the answer slot in a prompt, the model searches for the highest-scoring text $\hat{z}$ that maximizes the score of the language model as

$$\hat{z} = \underset{z \in \mathcal{Z}}{\text{search}} P(f_{\text{fill}}(x', z); \Theta). \tag{4.1}$$

The search function can be an *argmax* search, which looks for the output with the highest score, or sampling, which randomly generates outputs according to the LM probability distribution (P. Liu et al., 2023). If the model generates text, like T5 or BART, the search function can also be greedy or beam search.

The answer z comes from a set $\mathcal{Z}$ of permissible values. In the case of generative tasks, such as machine translation, the $\mathcal{Z}$ is the whole language. For classification tasks, $\mathcal{Z}$ is a small subset of words in the language (e.g. $\mathcal{Z} = \{$*"great"*, *"terrible"*$\}$ to represent classes in $\mathcal{Y} = \{+, -\}$).

After the highest-scoring answer $\hat{z}$ is found, it is mapped to the highest-scoring output $\hat{y}$. In the case of language generation, the answer itself is the output. For other tasks, like classification, the mapping is the inverse function used to create $\mathcal{Z}$ from $\mathcal{Y}$.

In some cases, one output can be mapped to multiple answers. For instance, multiple different words can be used to represent positive sentiment (e.g. *"great"*,

"excellent", *"amazing"*) to express a single class, in this case, "++". In such cases, the possible mapping to multiple answers must be considered when mapping the response back to the output.

The answer can also have different shapes (P. Liu et al., 2023), like a single token from a pre-trained language model's vocabulary, a short multi-token span, a sentence or a document.

4.2.2 Prompt Engineering

Selecting a prompt that leads to the most effective performance in a downstream task is called *prompt engineering*. When creating the prompt, the *prompt shape* and whether the template should be created manually (created intuitively based on human introspection) or automatically must be considered.

4.2.2.1 Prompt Shape

P. Liu et al. (2023) describe two shapes (also called types) of prompts: *prefix prompts* and *cloze prompts*. Prefix prompts have the unfilled slot in the middle of the text (e.g. "`[X]` It was a `[Z]` movie"). In cloze prompts, the input text comes entirely before the slot for the answer (e.g. "`[X]` It was `[Z]`").

Choosing the prompt shape depends on the task and model used. For generation tasks or tasks solved using autoregressive language models, prefix prompts are often more favourable because they fit well with the left-right nature of the models. On the other hand, cloze prompts are well suited for tasks solved using masked language models because they closely match the form of the pre-training objective. Both types of prompts can be used with full-text reconstruction models because they are more versatile. Figure 4.4 compares masked and autoregressive language models and which prompt shape is more suitable for each model type.

For some tasks involving multiple inputs, such as text pair classification, templates have to contain several slots for inputs (`[X1]`, `[X2]`, ...).

4.2.2.2 Automated Template Learning

Manually created templates have several issues. First, the creation requires lots of time and experience, especially for complex tasks such as semantic parsing. Secondly, finding optimal prompts might be impossible even for experienced prompt creators (Jiang et al., 2020).

Various methods have been proposed to automate the template design procedure to overcome these problems (P. Liu et al., 2023). The automatically generated prompts can be divided into *discrete prompts* and *continuous prompts*.

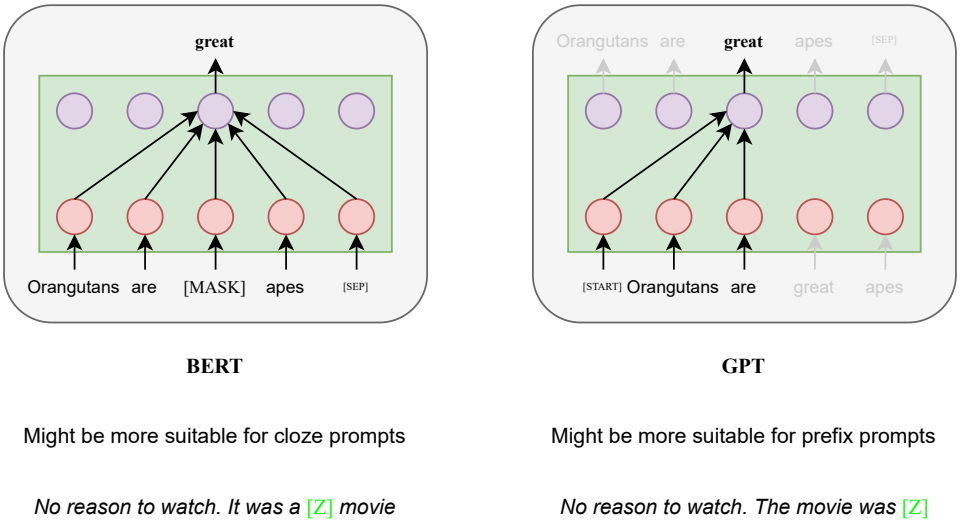


Figure 4.4: Comparison between a masked LM represented by BERT and an autoregressive LM represented by GPT, showing which prompt shape is more suitable for each model type, with examples.

4.2.2.3 Discrete Prompts

Discrete (hard) prompts use templates described in a discrete space, usually corresponding to actual natural language phrases. Several methods have been proposed for creating these prompts (P. Liu et al., 2023), some of which are described below.

Prompt Mining. The mining approach proposed by Jiang et al. (2020) searches a large text corpus, such as Wikipedia, for strings containing input x and output y and finds middle words or dependency paths between the input and outputs. Common middle words or dependency paths can be used in a template (e.g. "`[X]` It was `[Z]`").

Prompt Paraphrasing. Paraphrasing-based approaches, such as those explored by Jiang et al. (2020), involve paraphrasing an existing prompt (e.g. mined or manually created) into a set of alternative candidate prompts.

Prompt Generation. Gao et al. (2021) treat prompt generation as a task of generating text. As depicted in Figure 4.5, they use the T5 model to generate the templates. Then they fine-tune a specific model with these generated templates and select the best template based on the evaluation. The T5 model is suitable for template generation. As shown in Figure 3.14 in Chapter 3, the T5 model replaces randomly selected spans of tokens with unique placeholders and then reconstructs the masked spans. The authors also showed that automatically generated templates often outperform manually created templates, the demonstrations can improve the results, and prompt-based learning achieves better results than traditional fine-tuning when only a few examples are available (few-shot learning).

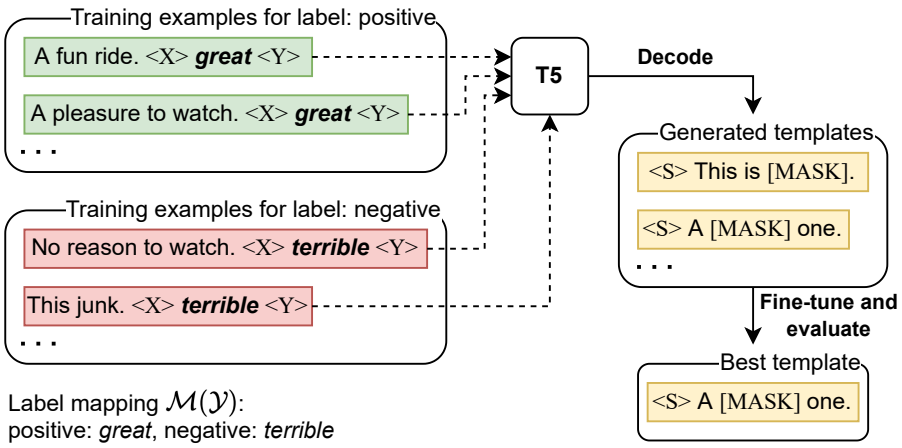


Figure 4.5: Template generation using the T5 model proposed by Gao et al. (2021).

4.2.2.4 Continuous Prompts

Continuous (soft) prompts differ from discrete prompts because they are described in the embedding space of the underlying language model rather than using actual natural language phrases. Continuous prompts remove the constraint that the embeddings of template words must correspond to embeddings of natural language (e.g. English) words. Instead, the template can be prefixed with pseudo tokens that are not used to construct the answer but are used to guide the model to predict the output. Continuous prompts allow the template's parameters to be fine-tuned based on the training data of downstream tasks, leaving the original parameters frozen. While this can reduce the training time and make the model easier to use for different tasks, continuous prompts usually require more training data than discrete prompts and are less suitable for few-shot learning (P. Liu et al., 2023).

Examples of continuous prompts include those explored by Li and Liang (2021) and Zhong et al. (2021). An example of the method proposed by Li and Liang (2021) is shown in Figure 4.6.

4.2.3 Multi-Prompt Learning

Multi-prompt learning methods use multiple prompts to improve the effectiveness of prompting methods (P. Liu et al., 2023). Figure 4.7 shows different multi-prompt learning strategies.

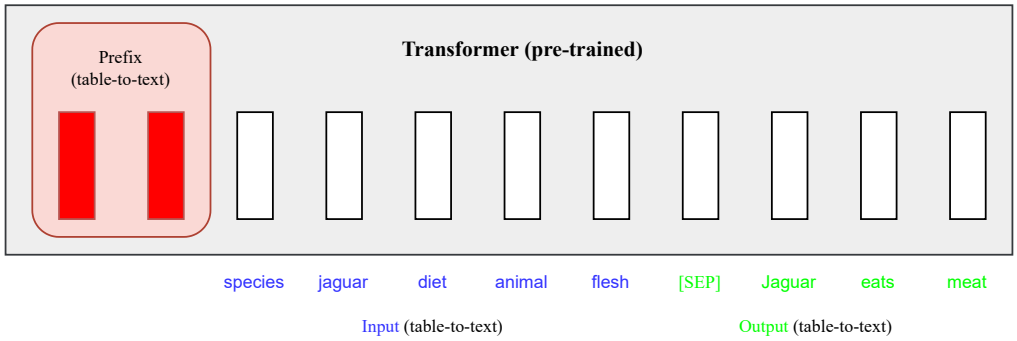


Figure 4.6: Prefix-tuning for a table-to-text task proposed by Li and Liang (2021). New embeddings are added to the pre-trained model. These new parameters (red prefix blocks) are then fine-tuned while the rest of the model parameters are frozen.

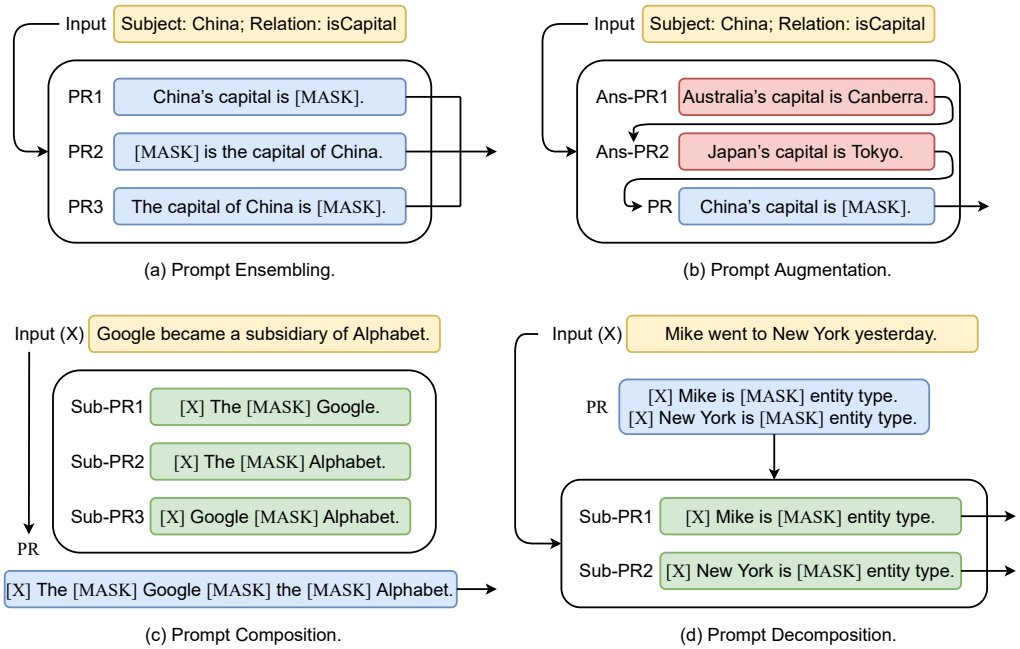


Figure 4.7: Different multi-prompt learning strategies. The yellow colour represents input text, blue prompt ("PR"), green sub-prompt ("Sub-PR"), and red answered prompt ("Ans-PR") (adapted from P. Liu et al., 2023).

4.2.3.1 Prompt Ensembling

Prompt ensembling uses multiple unanswered prompts to take advantage of the complementary benefits of different prompts. It can also reduce the cost of prompt designing and stabilize the performance on downstream tasks (P. Liu et al., 2023). The predictions from different prompts can be combined, for instance, using uniform or weighted averaging. Figure 4.7-(a) shows an example of prompt ensembling.

4.2.3.2 Prompt Augmentation

Prompt augmentation (Gao et al. (2021) called it *demonstration learning*) provides a few additional answered prompts to lead the model to correct predictions. Figure 4.7-(b) shows an example of prompt augmentation.

4.2.3.3 Prompt Composition

Prompt composition can be used on complex tasks based on more basic subtasks by combining sub-prompts for each sub-task into one prompt (P. Liu et al., 2023). Figure 4.7-(c) illustrates this process.

4.2.3.4 Prompt Decomposition

For tasks where multiple predictions need to be made for a single sample (e.g. sequence labelling), defining one prompt concerning the entire input text x can be challenging. *Prompt decomposition* solves this problem by breaking one prompt into several sub-prompts that can be answered separately, as shown in Figure 4.7-(d) for the named entity recognition task.

Related Work 5

This chapter briefly reviews the literature related to the thesis topic, specifically aspect-based sentiment analysis, cross-lingual sentiment analysis and cross-lingual aspect-based sentiment analysis. It is important to note that while some authors use the same datasets, their results are often not comparable due to differences in how they define and solve ABSA tasks. Additionally, they may modify the datasets slightly, such as adding or changing annotations or removing examples.

Zhang, Li et al. (2021) introduce GAS, a generative framework for aspect-based sentiment analysis. They propose two paradigms to formulate ABSA tasks as text generation problems: annotation-style modelling and extraction-style modelling. In the annotation-style modelling paradigm, the authors add annotations to a given sentence to incorporate label information in the target sentence. In contrast, the extraction-style modelling paradigm uses the desired natural language label of the input sentence as the target. The authors evaluate four ABSA tasks (UABSA, AOPE, TASD and ASTE) using the base T5 model for text generation. They also employ a prediction normalization strategy to fix misspellings and typos (e.g. *"Bbq ribs"* to *"BBQ ribs"*). The experiments are performed on four SemEval datasets: *Laptop14* (Pontiki et al., 2014), *Rest14* (Pontiki et al., 2014), *Rest15* (Pontiki et al., 2015) and *Rest16* (Pontiki et al., 2016). The authors demonstrate that their proposed approach for transforming ABSA tasks into text-generation problems is effective, as evidenced by the results for each task in Table 5.

Task	Lap14	Rest14	Rest15	Rest16
AOPE	69.55	75.15	67.93	75.42
UABSA	68.64	77.13	66.78	73.64
ASTE	60.78	72.16	62.10	70.10
TASD	-	-	61.47	69.42

Table 5.1: Best results of the methods proposed by Zhang, Li et al. (2021) for different tasks and datasets. F1 scores are reported.

Zhang, Deng et al. (2021) address aspect-based sentiment analysis by proposing a new task called aspect sentiment quad prediction (ASQP). The task involves predicting all (aspect category c, aspect term a, opinion term o, sentiment polarity p) quads for a given sentence. They treat this as a sequence-to-sequence problem and employ the T5-base model to solve it. To transform a sentiment quad $q = (c, a, o, p)$ to a natural sentence, they use the projection function $P_z(\cdot)$ for $z \in \{c, a, o, p\}$ and map the quad to a sentence of the form *"$P_c(c)$ is $P_p(p)$ because $P_a(a)$ is $P_o(o)$."*. For example, a quad (*food quality*, *steak*, *undercooked*, *negative*) can be transformed into a sentence *"Food quality is bad because steak is undercooked"*. If multiple such sentences exist for a given input, they concatenate them using the `[SSEP]` token to form the final target sequence y. Their method, `PARAPHRASE`, can also solve other ABSA tasks, such as aspect sentiment triplet extraction (ASTE) and target-aspect-category detection (TASD). To evaluate their method, the authors create new datasets *Rest15* and *Rest16* by augmenting the original *Rest15* and *Rest16* SemEval datasets (Pontiki et al., 2015; Pontiki et al., 2016) with an opinion term (e.g. *"delicious"*) for annotated examples. Each sample in these datasets comprises a review sentence and one or more sentiment quads. The authors report F1 scores, precision and recall on both datasets, considering a prediction correct only if all the predicted elements match the gold labels exactly. Table 5 summarizes the results for different tasks.

Task	**Rest15**	**Rest16**
ASTE	62.56	71.70
TASD	63.06	71.97
ASQP	46.93	57.93

Table 5.2: F1 scores for different tasks and datasets achieved by the `PARAPHRASE` method proposed by Zhang, Deng et al. (2021).

Zhang, He et al. (2021) focus on cross-lingual aspect-based sentiment analysis and present an *alignment-free* label projection method. This approach eliminates the need for word alignment tools to project labels from the source sentence into the translated target sentence. First, the authors mark each aspect term in a sentence with a special symbol, such as brackets and braces. They then use the Google Translate API[1] to translate the sentence, which preserves more task-specific knowledge. The aspect terms in the translated sentence can be extracted by special symbols. The authors also introduce an aspect code-switching (ACS) mechanism, which creates two bilingual sentences by switching the aspect terms between the source and translated target sentences. Figure 5.1 illustrates the aspect code-switching mechanism

[1] `https://translate.google.com/`

and *alignment-free* method. The authors fine-tune models, specifically the multilin-

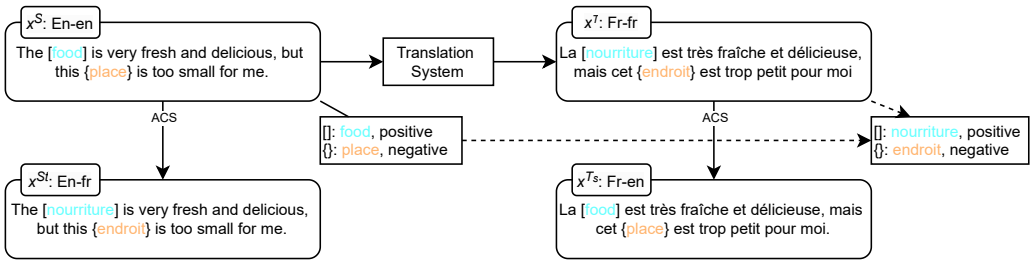


Figure 5.1: Example of the aspect code-switching technique (lower part) and *alignment-free* label projection method (upper part) (adapted from Zhang, He et al., 2021).

gual BERT and the base XLM-RoBERTa, on a combination of the code-switched bilingual sentences and the monolingual sentences of the source/target languages. The models aim to predict the aspect terms and their corresponding sentiment polarities. The authors evaluate the effectiveness of their methods through experiments with five languages from the SemEval-2016 dataset (Pontiki et al., 2016) (specifically the *Rest16* dataset). Table 5 shows the results.

mBERT				**XLM-R**			
fr	es	nl	ru	fr	es	nl	ru
52.25	62.91	53.40	54.58	61.00	69.24	63.74	62.02

Table 5.3: Cross-lingual performance of two models proposed by Zhang, He et al. (2021), fine-tuned on English as the source language, across various target languages. F1 scores are reported.

Gao et al. (2021) focus on regression and classification tasks, including sentiment analysis, using prompt-based fine-tuning. They demonstrate that with a few training examples, prompt-based fine-tuning outperforms traditional fine-tuning. Furthermore, they show that adding demonstrations to a prompt is an effective fine-tuning method, and their automatic prompt search method performs as well as or better than manual prompts. In their experiments on various datasets, the authors use the T5 model to automatically generate prompts and the BERT-large and RoBERTa-large models for fine-tuning. Chapter 4) provides more details on their method.

Gao et al. (2022) propose `LEGO-ABSA`, a unified generative multi-task framework for solving multiple ABSA tasks simultaneously. Instead of treating the prompt and

output text as simple strings, they consider them as a combination of multiple elements to be extracted. The T5 model is used in this approach. The authors introduce element prompts for aspect, category, opinion and sentiment, which can be concatenated to form the final task prompt, where every sentinel token has a unique ID. For aspect, it has the following form: "aspect: <extra_id_*id*>", where *id* is the ID of the sentinel token. The model can be trained on multiple ABSA tasks simultaneously using different task prompts (e.g. one sentence with element prompts for aspect and opinion and a second sentence with element prompts for aspect and sentiment) and can thus solve multiple tasks simultaneously. The proposed method is tested on several SemEval datasets (Pontiki et al., 2015; Pontiki et al., 2014; Pontiki et al., 2016), achieving state-of-the-art results on multiple tasks. Table 5 summarizes the results obtained for different tasks and datasets.

Task	**Lap14**	**Lap15**	**Lap16**	**Rest14**	**Rest15**	**Rest16**
AOPE	71.3	-	-	78.1	72.9	77.6
E2E-ABSA	72.3	-	-	80.6	74.3	78.6
ASTE	62.2	-	-	73.7	64.4	71.5
ACSA	-	65.0	55.9	-	71.0	76.2
TASD	-	-	-	-	62.3	71.8
ASQP	-	-	-	-	46.1	57.7

Table 5.4: Best results of the methods proposed by Gao et al. (2022) for different tasks on different datasets. F1 scores are reported.

Přibáň et al. (2022) deal with cross-lingual sentiment analysis in English, Czech and French. They perform cross-lingual classification without using any labelled data in the target language (zero-shot) using LSTM and CNN classifiers, along with five linear transformations. The authors pre-train embeddings from the target domain and show that these are important for improving the cross-lingual sentiment analysis results. Besides comparing individual linear transformations, they also demonstrate that the transformation-based methods can compete to some extent with state-of-the-art multilingual models based on the BERT architecture.

Experiments 6

This chapter provides a detailed description of the experiments performed in this thesis. First, the datasets used in the experiments are described. Then the problem, solution and evaluation methods used in this thesis are explained. Finally, the results of the experiments are presented and discussed in detail.

6.1 Datasets

We use datasets provided by Pontiki et al. (2016) for the experiments. Datasets comprise actual reviews of restaurants written by users in six languages: English (en), Spanish (es), French (fr), Dutch (nl), Russian (ru) and Turkish (tr). Datasets are already split into training and testing sets for each language. We keep the original split and further sample 10% data from the training set as the validation set used for model selection. Each sample sentence is annotated with sentiment triplets (aspect category, aspect term, sentiment polarity), although some sentences may have no annotations. Table 6.1 shows the data statistics for each language in datasets.

		en	es	fr	nl	ru	tr
Train	Sentences	2000	2070	1733	1711	3490	1104
	Triplets	2507	2720	2530	1860	4022	1535
Test	Sentences	676	881	696	575	1209	144
	Triplets	859	1072	954	613	1300	155

Table 6.1: Statistics of the data in each language.

6.1.1 New Czech Dataset

To evaluate our methods on the Czech language, we present a new dataset of restaurant reviews in Czech using data from Hercig et al. (2016). The original dataset does not have the required annotation format for subtask 1 of SemEval 2016 Task

5 solved in this thesis. Therefore, we keep the original data and add the required annotations to make the dataset compatible with this task. Two annotators contributed to the annotation of the new dataset: the author and the supervisor of this thesis, both native speakers with experience in sentiment analysis.

Following Pontiki et al. (2016), we measure the inter-annotator agreement (IAA) between the two annotators in terms of micro F1 score for the identification of aspect terms, aspect categories, (aspect term, aspect category) tuples, and (aspect term, aspect category, polarity) triplets. Table 6.2 shows the inter-annotation agreement results, which we consider high enough for all annotation targets.

Annotation target	**IAA**
Aspect term	93.19
Aspect category	93.00
Aspect term & aspect category	91.06
Aspect term & aspect category & polarity	89.70

Table 6.2: Inter-annotator agreement (IAA) for different annotation targets in the new Czech dataset, measured in terms of micro F1 score (in %).

We split the dataset into training and test splits in a 75:25 ratio. The data statistics, including the number of sentences and the number of annotation triplets, are shown in Table 6.3.

Split	Sentences	Triplets
Train	1612	2764
Test	537	907

Table 6.3: Data statistics of the new Czech dataset.

6.2 Problem Statement

This thesis solves the first subtask (SB1) proposed by Pontiki et al. (2016), which is the sentence-level ABSA. The goal of this task is to identify all opinion tuples with the following tuple slots:

1. Aspect category (slot 1): Identifying the category of the aspect that the opinion refers to (ACD task).

2. Opinion target expression (slot 2): Extracting the specific aspect term or phrase the opinion refers to (ATE task).

3. Aspect category and corresponding opinion target expression (slot 1&2): Finding the aspect category and the specific aspect term or phrase to which the given opinion refers.

4. Sentiment polarity (slot 3): Determining the sentiment of the opinion for a given (aspect category, aspect term) pair.

Figure 6.1 shows an example of an opinion tuple for a given sentence. More details are described in Chapter 2.

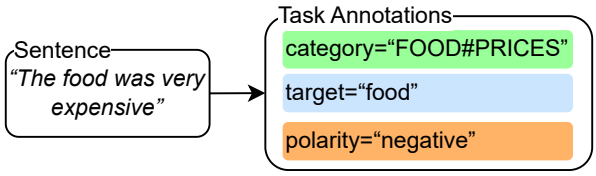


Figure 6.1: Example of a sentence with annotations for the task solved in this thesis.

In addition, we solve the target-aspect-sentiment detection (TASD) task, which involves identifying the aspect category with the corresponding aspect term and sentiment polarity. Given a sentence x, target-aspect-sentiment detection aims to predict all aspect level sentiment triplets $\{(c, a, p)\}$, where c is aspect category, a aspect term and p sentiment polarity. The aspect category belongs to the set of categories V_c. The aspect term is usually a text in a sentence. However, it can also be "NULL" if the target is not explicitly mentioned in a text: $a \in V_x \cup \{\emptyset\}$, where V_x refers to the set containing all possible continuous ranges of x. The sentiment polarity belongs to one of the sentiment classes $\{negative, positive, neutral\}$. Similarly, we can define the tasks of identifying slot 1, slot 2 and slot 1&2 tuples, where the aim is to return all $\{c\}$, $\{a\}$ and $\{(c, a)\}$ tuples, respectively.

6.3 Sequence-to-Sequence Models

As our solution, we propose a unified sequence-to-sequence method to solve multiple tasks simultaneously using sequence-to-sequence models. The tasks we address are ATE (extracting individual aspect terms, slot 2), ACD (extracting individual aspect categories, slot 1), extracting pairs of aspect terms and categories (slot 1&2) and TASD, which involves extracting (aspect term, aspect category, sentiment polarity) triplets. This approach can be used with traditional fine-tuning (TR-FT) and prompt-based fine-tuning (PT-FT).

Initially, we incorporated all training examples, including those without sentiment triplets, in our experiments. For such examples, both the input and label were empty strings. The results of these examples are shown in Appendix B (see Table B.1

for results on the English dataset, Table B.2 for monolingual results and Table B.3 for cross-lingual results). However, following related work (Gao et al., 2022; Zhang, Deng et al., 2021; Zhang, Li et al., 2021), the final experiments are conducted only on samples with sentiment triplets to allow better comparison.

6.3.1 Models for Traditional Fine-Tuning

Our approach for traditional fine-tuning involves converting sentiment triplets (aspect term, aspect category, sentiment polarity) from a given sentence into a natural language sentence. For example, given the input sentence *"Steak was amazing"*, we extract the sentiment triplet (*food quality*, *steak*, *positive*), which we then transform into a sentence *"Food quality is great, given the expression: steak"*. We fine-tune a sequence-to-sequence model, specifically the T5 and BART models and their multilingual versions (mT5 and mBART), using the input sentence and transformed triplet. This approach is inspired by Zhang, Deng et al. (2021), where the authors use a different form of transformed triplets, use only the T5 model, and evaluate it only on the English dataset.

6.3.1.1 Label Construction

This section describes the creation of labels from sentiment triplets. We convert each sentiment triplet $t = (c, a, p)$ into a natural language sentence using a projection function $P_z(\cdot)$ for each sentiment element $z \in \{c, a, p\}$ as follows:

$$\text{"}P_c(c) \text{ is } P_p(p), \text{ given the expression: } P_a(a)\text{"}.$$

The projection function maps sentiment elements into a natural language form. Since the aspect term a is already in a natural language form, we use $P_a(a) = a$. The category in the data we use is in `E#A` format, where `E` is the entity type, and `A` is the attribute type. We split the entity type and attribute type and convert all uppercase letters into lowercase letters except for the first letter in the entity type. For example, `FOOD#QUALITY` becomes *"Food quality"*. For the sentiment polarity, the projection function is

$$P_p(p) = \begin{cases} great & \text{if } p \text{ is } positive, \\ ok & \text{if } p \text{ is } neutral, \\ bad & \text{if } p \text{ is } negative. \end{cases} \tag{6.1}$$

Finally, we concatenate all transformed triplets using *";"* to create the final target sentence y. Figure 6.2 shows how we construct the target sentence from an input sentence with sentiment triples.

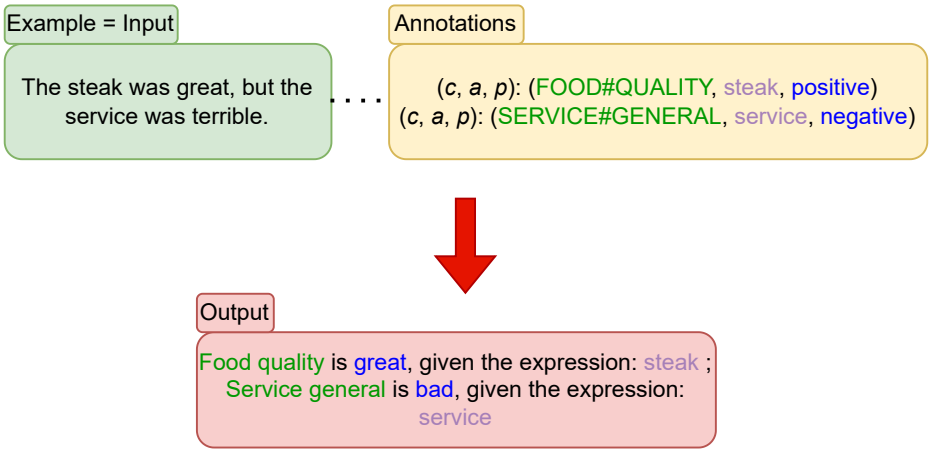


Figure 6.2: Example of the target sentence construction for sequence-to-sequence models from an input sentence with annotated sentiment triplets using traditional fine-tuning.

6.3.2 Prompting Models

We propose a prompt-based method of solving the ABSA task in addition to the traditional fine-tuning method. The method involves expanding the example sentence x with a template t to create the final input x': $x' = x + | + t$. To generate the template, we use the same method of transforming the sentiment triplets into natural language sentences as described earlier ("$P_c(c)$ *is* $P_p(p)$*, given the expression:* $P_a(a)$"). This method is inspired by Gao et al. (2022), who use a different prompt design. However, their method is limited to the T5 model and was evaluated only on the English dataset. The number of transformed triplets in the prompt corresponds to the number of triplets provided for one example, giving the prompting models a possible advantage over the traditionally fine-tuned models that do not have any information about the number of triplets for an example in advance. Since the training objective of the (m)T5 and (m)BART models differ, we need to design the prompt differently for each group of models.

6.3.2.1 T5 Model

The T5 model aims to reconstruct randomly selected continuous spans of input text that are masked by sentinel tokens <extra_id_*id*> during pre-training. Here, *id* refers to the ID of the sentinel token, which starts from zero and increments by one. The model replaces non-masked spans of text with sentinel tokens. In our method, we replace the aspect category with <extra_id_0>, the sentiment polarity with <extra_id_1>, and the aspect term with <extra_id_2> to create the final input. The output of the T5 model consists of the aspect category, sentiment polarity and

aspect term separated by sentinel tokens. Figure 6.3 shows how we construct the input and output from the original input text with sentiment triplets for the T5 model.

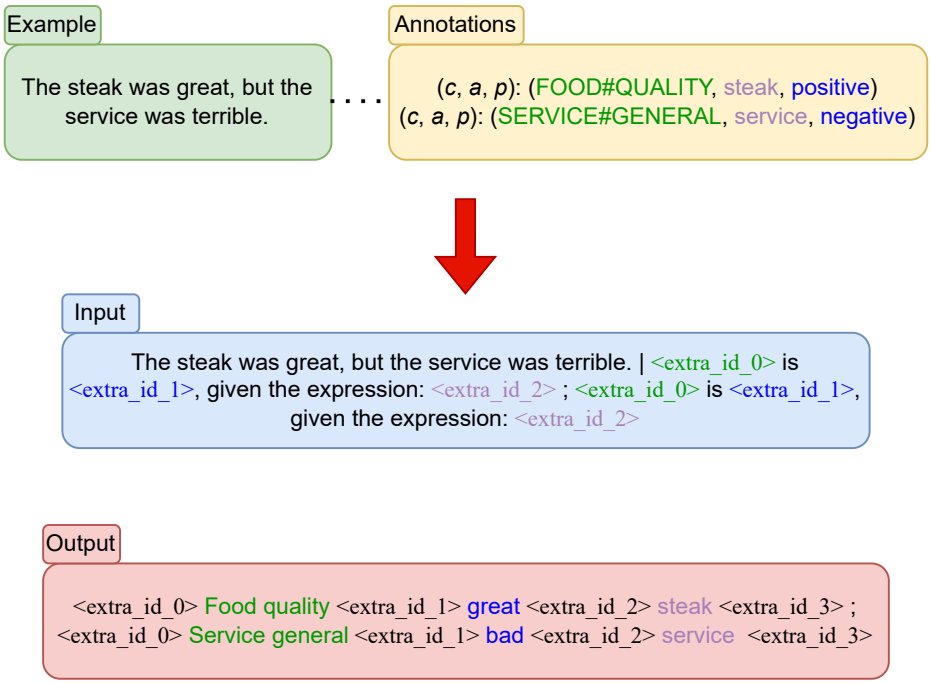


Figure 6.3: Example of the input and output construction from one example with sentiment triplets for the T5 model with prompting.

6.3.2.2 BART Model

The objective of the BART model is distinct from that of the T5 model. Unlike T5, BART aims to reconstruct the entire input text rather than just masked spans. Furthermore, the BART model utilizes the `<mask>` token instead of sentinel tokens. Figure 6.4 illustrates how we construct the input and output from the original input text with sentiment triplets for the BART model.

6.4 Models for Sentiment Polarity Classification

This section describes the method we propose for models used for the sentiment polarity classification given the aspect term and aspect category (slot 3 in Pontiki et al., 2016). This method can also be used with traditional and prompt-based fine-tuning. Our proposed method for sequence-to-sequence models cannot produce comparable results for this task because the models have no prior information about

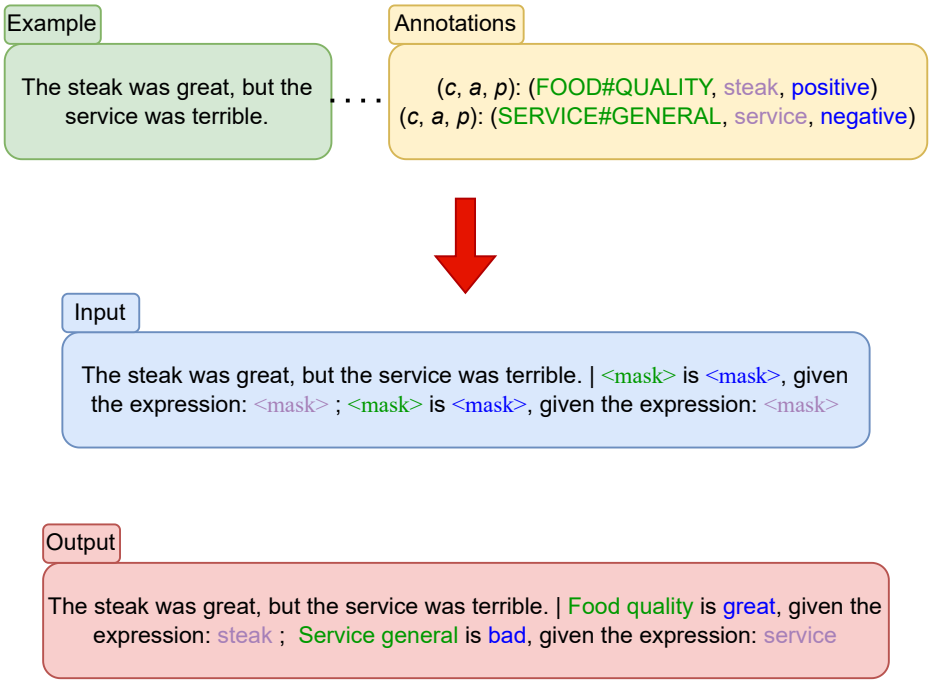


Figure 6.4: Example of the input and output construction from one example with sentiment triplets for the BART model with prompting.

the aspect term and category and attempt to predict them along with the sentiment. For comparable results, we would need to change the input, output and prompts for these models.

6.4.1 Models for Traditional Fine-Tuning

During traditional fine-tuning, we create a single input in the form of $x + | + P_c(c) + a$ for each aspect term a and aspect category c in the annotation triplets for a single example sentence x, where $P_c(c)$ is the same projection function for aspect category that we use in sequence-to-sequence models. Figure 6.5 shows an example of such input construction with desired outputs. We then feed this input to a model with a classification head to obtain the sentiment. For this task, we experiment with BERT, RoBERTa, multilingual BERT, XLM-RoBERTa and ELECTRA models (including prompting, discussed in the following subsection).

6.4.2 Prompting Models

In our classification model, we experiment with prompting, similar to what we do with sequence-to-sequence models. For each aspect term a and aspect category c in annotation triplets for a single example sentence x, we create a single input in the

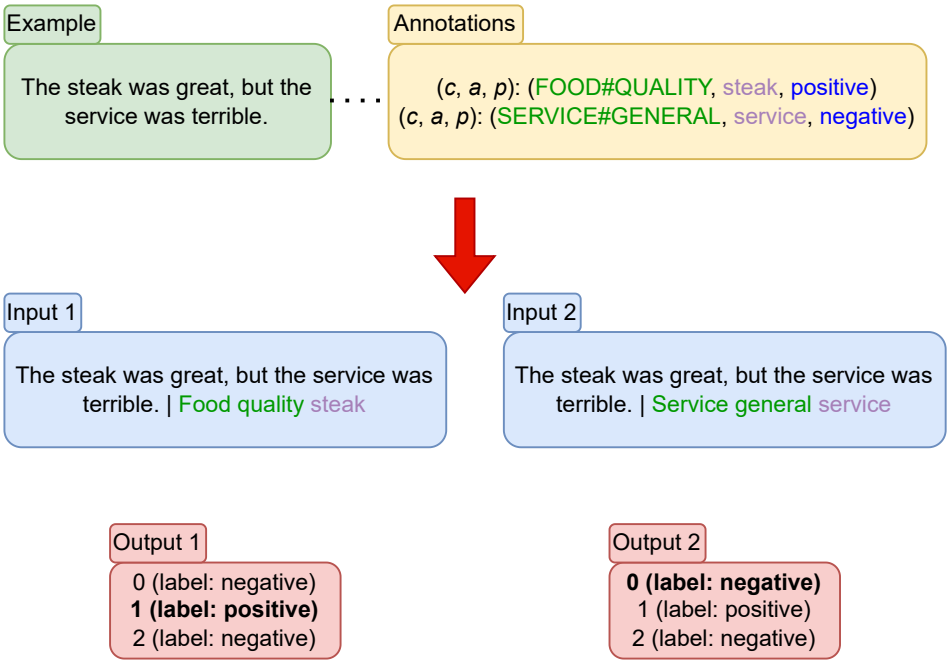


Figure 6.5: Example of the input construction from one example text with sentiment triplets and the outputs for the classification model using traditional fine-tuning.

form of $x + | + t$, where t is the sentiment triplet transformed into a natural language sentence with one empty slot for the sentiment. We use the same transformation function as in sequence-to-sequence models (*"$P_c(c)$ is $P_p(p)$, given the expression: $P_a(a)$"*). Equation 6.1 shows the projection function for the sentiment polarity p. The polarity is replaced by the `[MASK]` token in inputs. Figure 6.6 shows an example of input construction with desired outputs. This method is inspired by Gao et al. (2021), where the authors use different prompts for different tasks using sentiment classification models.

The model returns probabilities for each token in its tokenizer vocabulary for the position of the `[MASK]` token. We take the token with the highest probability as the predicted word. During the training, we optimize the cross-entropy loss (the loss is optimized only for the position of the `[MASK]` token, not all inputs).

6.5 Evaluation

This section defines some evaluation metrics used in this thesis and describes the evaluation process of our proposed models.

The simplest and most commonly used metric is **accuracy**, defined as the number of correctly predicted samples divided by the total number of samples (see Equation 6.2). However, this metric is not suitable for training sets that have an

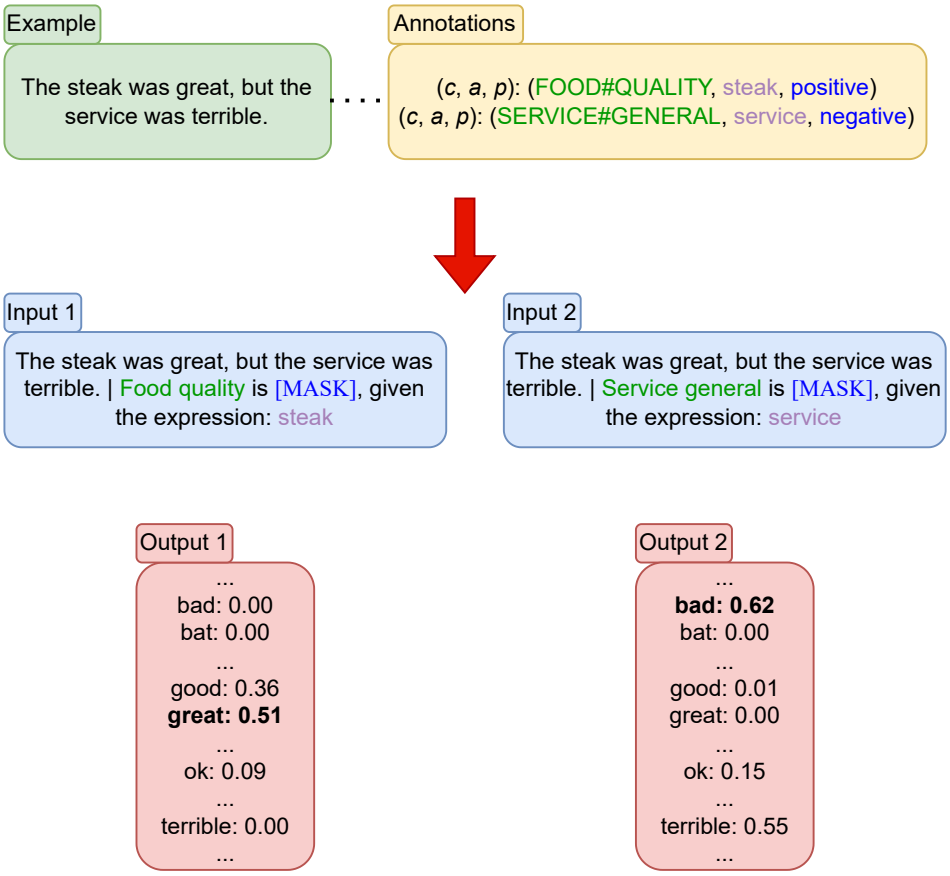


Figure 6.6: Example of the input construction from one example text with sentiment triplets and the outputs for the classification model using prompting.

unbalanced number of classes. For instance, if 90% of the samples in the training set belong to class 1, the classifier can achieve a 90% success rate by simply classifying all data as class 1. Furthermore, accuracy is often too strict for multi-label classification, where each sample can be assigned multiple labels. Therefore, it is more appropriate to use other metrics in some cases.

$$\text{accuracy} = \frac{\text{correctly classified samples}}{\text{total number of samples}} \tag{6.2}$$

It is necessary to explain the concepts of **true positive** (TP), **true negative** (TN), **false positive** (FP) and **false negative** (FN) to define other metrics. These results can be formulated in a confusion matrix (see Table 6.4).

Precision is the fraction of relevant samples among the retrieved samples (see Equation 6.3). In other words, it measures how many of the samples the model classified as positive are actually positive. **Recall** is the fraction of retrieved relevant samples (see Equation 6.4). In other words, it measures how many actual positive

		True class	
Predicted class		1	0
	1	TP	FP
	0	FN	TN

Table 6.4: Confusion matrix.

samples the model correctly identified as positive. Finally, the **F1 score** is the harmonic mean of precision and recall (see Equation 6.5) (Manning et al., 2008).

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.3}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.4}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.5}$$

The above equations are valid only for binary classification (classification into two classes). For multi-class (classification into K classes) and multi-label classification, micro, macro, and weighted averages are used (Manning et al., 2008). Let $B(\text{tp}, \text{fn}, \text{fp}, \text{tn})$ be a given metric computed based on the number of TP, FN, FP and TN. Micro-averaged metrics are computed from the sum of TP, TN, FP and FN over all classes (see Equation 6.6). It holds that the results of all three micro-averaged metrics (recall, precision and F1 score) and accuracy are the same for the multi-class classification. Macro-averaged and weighted-averaged metrics are calculated separately for each class. Then the arithmetic mean is calculated from these values in the first case (see Equation 6.7) and the weighted average in the second case (see Equation 6.8, where f_k is the number of samples belonging to class k divided by the number of all samples). Table 6.5 shows an example of a confusion matrix for three classes.

$$B_{\text{micro}} = B\left(\sum_{k=1}^{K} \text{tp}_k, \sum_{k=1}^{K} \text{fn}_k, \sum_{k=1}^{K} \text{fp}_k, \sum_{k=1}^{K} \text{tn}_k\right) \tag{6.6}$$

$$B_{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} B(\text{tp}_k, \text{fn}_k, \text{fp}_k, \text{tn}_k) \tag{6.7}$$

$$B_{\text{weighted}} = \sum_{k=1}^{K} f_k B(\text{tp}_k, \text{fn}_k, \text{fp}_k, \text{tn}_k) \tag{6.8}$$

	True class		
Predicted class	orca	dolphin	whale
orca	10	1	3
dolphin	1	7	0
whale	4	0	6

Table 6.5: Example of a confusion matrix for three classes. For example, it can be seen that out of a total of 15 photos of an orca, the classifier incorrectly identified the class as a total of five times (four times as a dolphin and once as a whale).

6.5.1 Evaluation of Proposed Models

We follow the evaluation process described by Pontiki et al. (2016) to evaluate our proposed models for the tasks we solve.

For sequence-to-sequence models, we report four numbers. For slot 1 (aspect category detection), slot 2 (aspect term extraction) and the slot 1&2 tuples, we calculate the micro F1 scores by comparing the returned annotations with the gold annotations. We ignored duplicate categories to calculate the micro F1 score for slot 1. For slot 2, only distinct targets are considered, and "NULL" targets are discarded. We also report the micro F1 score for the TASD task.

To avoid any impact on the results, we count the triplets and (aspect term, aspect category) tuples generated multiple times as one for the sequence-to-sequence models.

For the classification models, we report accuracy for slot 3 (sentiment polarity classification), defined as the number of correctly predicted polarity labels of the gold aspect categories divided by the total number of the gold aspect categories.

6.6 Experiments Details

This section provides details on the experiments conducted in this thesis. For the sequence-to-sequence models, we use the large T5 and large BART models for the monolingual experiments on the English dataset. For cross-lingual and monolingual experiments on all languages, we use the large multilingual T5 model (**mT5**) (Xue et al., 2021) and the large multilingual BART model (**mBART**) (Tang et al., 2020). Regarding the training, we use the batch size of 64 (16, 32 and 128 lead to similar performances). For all models, we experiment with learning rates of 1e-4, 3e-4, 1e-5 and 5e-5. We use the AdaFactor optimizer (Shazeer & Stern, 2018) for m(T5) models and the AdamW optimizer (Loshchilov & Hutter, 2019) for (m)BART models. We experiment with greedy and beam search decoding with the beam size of 3 and 5

for generating the output sequence. All lead to similar results. Therefore, we use greedy search decoding for simplicity. The number of training epochs is 35 for all experiments. We track the best score on the validation data for the TASD task to select the best-performing model, including selecting the best number of epochs. Tracking the score for the TASD task led to slightly better results than tracking the validation loss.

For the sentiment polarity classification models, we use the large uncased BERT model (Devlin et al., 2019), large RoBERTa model (Y. Liu et al., 2019), and large ELECTRA model (Clark et al., 2020) for the monolingual experiments on the English dataset. For cross-lingual experiments and monolingual experiments on all languages, we use the uncased multilingual BERT model (**mBERT**) (Devlin et al., 2019) and the large XLM-RoBERTa model (**XLM-R**) (Conneau et al., 2020). We use the batch size of 64 (16, 32 and 128 lead to similar performances) and 50 training epochs for all experiments. We use the AdamW optimizer for all models and experiment with learning rates of 1e-4, 5e-5 and 1e-5. We choose the best-performing model based on the slot 3 performance on the validation data, including the number of epochs.

We utilize the sequence length for all models to prevent truncation of inputs and labels (if they are sequences). Additionally, we confirm that all samples fit into the models as none exceed 512 tokens.

The monolingual models are fine-tuned on the training data and evaluated on the validation data of the same language. After selecting the best model parameters, such as learning rate and the number of epochs, we fine-tune the model on all the original training data (our training and validation data combined) and evaluate the model on the test data.

All of the cross-lingual experiments are done in a zero-shot setting (known as *zero-shot cross-lingual*), which involves evaluating a model on data from *target language* without using annotated data from that language during training. The model is trained on data from another language called *source language*.

The cross-lingual models are fine-tuned on all the original training data of the source language and evaluated on the validation data of the target language. Afterwards, the best-performing model is evaluated on the test data of the target language. These models perform similarly to those where source language was used for both training and validation data.

We run the best-performing models five times with different random seed initialization and report the 95% confidence interval (CI) for the mean defined as

$$\mathrm{CI} = \bar{x} \pm t \cdot \frac{s}{\sqrt{n}}, \tag{6.9}$$

where $\bar{x}$ is the sample mean, s sample standard deviation, n sample size (in our case, 5) and t t-value for the confidence level.

6.6.1 Number of Parameters

Table 6.6 shows the number of parameters of models used in this thesis when using traditional and prompt-based fine-tuning.

Model	TR-FT	PT-FT
Sequence-to-sequence models		
BART	406M	406M
T5	738M	738M
mBART	611M	611M
mT5	1.2B	1.2B
Models for polarity classification		
BERT	335M	335M
ELECTRA	335M	51M
RoBERTa	355M	355M
mBERT	167M	167M
XLM-R	560M	560M

Table 6.6: Number of parameters for models used in this thesis when using traditional fine-tuning (TR-FT) and prompt-based fine-tuning (PT-FT).

6.7 Implementation

The program developed for this thesis is implemented in Python 3.10.7, an interpreted language often used in data science projects. The program uses various libraries, such as *PyTorch* (Paszke et al., 2019), *PyTorch Lightning* (Falcon & The PyTorch Lightning team, 2019), *TorchMetrics* (Detlefsen et al., 2022), *WandB* (Biewald, 2020), and *Hugging Face's transformers* (Wolf et al., 2020). *PyTorch* and *PyTorch Lightning* are used to build and train deep learning models, while *TorchMetrics* provides useful metrics for evaluating their performance. *WandB* is used for logging and tracking experiments, and the *transformers* library provides state-of-the-art pre-trained language models that can be fine-tuned for specific tasks.

6.8 Results

This section provides an overview and evaluation of the results achieved from the experiments conducted in this thesis. It consists of three subsections. The first subsection discusses the results obtained from the sequence-to-sequence models, while

the second deals with the models used for sentiment polarity classification. Each subsection presents the cross-lingual and monolingual results of models with (prompt-based fine-tuning) and without (traditional fine-tuning) prompting. Additionally, each of these two sections compares the monolingual results of each dataset with the state-of-the-art results. Furthermore, they also compare the cross-lingual results with the monolingual results for a given target language. The last subsection evaluates the results overall.

6.8.1 Sequence-to-Sequence Results

This subsection presents the results obtained with our sequence-to-sequence method that solves the slot 1, slot 2, and slot 1&2 problems in the first subtask of the task proposed by Pontiki et al. (2016), as well as the TASD task.

6.8.1.1 Result on the English Dataset

Table 6.7 shows the results of our method with different models on the English dataset and compares them with the state-of-the-art results. Our proposed approach outperforms the previous state-of-the-art results (all previous state-of-the-art results have been obtained with base models, but we use large versions).

We observed that prompt-based fine-tuning outperforms traditional fine-tuning in all reported models except for the BART model. The reason might be that the models with prompting have dedicated slots for every annotation triplet in the prompt, which may give them an advantage over the traditional fine-tuned models that have no prior knowledge about the correct number of triplets for each sample. Additionally, it is possible that prompt-based fine-tuning is better suited for these models than the traditional approach. However, we noticed that the BART model struggles the most with generating the output in the correct format, and it is even worse when using prompting. Generating the output in the correct format is crucial because the evaluation process requires output in the given format.

Furthermore, the mBART model performs better than the BART model, even though the latter is pre-trained only on English data while the former is pre-trained on data from 50 languages. On the other hand, the T5 model performs better than the mT5 model. The performance of the mT5 model significantly drops when compared to the T5 model when using traditional fine-tuning. Upon examining the outputs produced by the mT5 model, we noticed that the model often generates repetitive outputs and produces the same (aspect category, aspect term) tuple with different polarity more often than other models (e.g. the model produces only *"Food quality is great, given the expression: steak"*, but multiple times, or for the (`FOOD#QUALITY`, steak) pair, it produces both *neutral* and *positive* sentiment).

Model	Slot 1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning				
T5	**86.57**$^{\pm 0.94}$	84.60$^{\pm 0.82}$	77.80$^{\pm 0.65}$	73.33$^{\pm 0.40}$
mT5	77.04$^{\pm 1.85}$	69.20$^{\pm 2.17}$	59.81$^{\pm 2.32}$	53.38$^{\pm 2.49}$
BART	78.65$^{\pm 0.91}$	71.55$^{\pm 0.70}$	66.56$^{\pm 0.78}$	62.41$^{\pm 0.90}$
mBART	80.64$^{\pm 1.68}$	80.82$^{\pm 0.94}$	70.15$^{\pm 1.91}$	62.82$^{\pm 1.95}$
Prompt-based fine-tuning				
T5	**88.57**$^{\pm 1.24}$	**87.98**$^{\pm 0.61}$	**80.77**$^{\pm 1.07}$	**76.15**$^{\pm 0.41}$
mT5	**87.20**$^{\pm 0.66}$	**86.85**$^{\pm 0.76}$	**79.12**$^{\pm 0.57}$	73.59$^{\pm 0.47}$
BART	78.89$^{\pm 0.41}$	59.91$^{\pm 0.79}$	58.37$^{\pm 0.27}$	53.94$^{\pm 0.14}$
mBART	84.56$^{\pm 0.74}$	84.88$^{\pm 1.15}$	75.34$^{\pm 0.98}$	69.82$^{\pm 1.21}$
State-of-the-art				
LEGO-ABSA (Gao et al., 2022)	-	-	-	71.80
PARAPHRASE (Zhang, Deng et al., 2021)	-	-	-	71.97
GAS (Zhang, Li et al., 2021)	-	-	-	69.42
NLANG (Pontiki et al., 2016)	73.03	72.34	52.61	-

Table 6.7: Micro F1 scores (in %) achieved by the sequence-to-sequence models on the English dataset with traditional and prompt-based fine-tuning compared to the state-of-the-art results. Each task's best result (or results in case of overlapping confidence intervals) is highlighted in **bold**.

On the other hand, when using prompting, the mT5 model is only slightly worse than the T5 model. Overall, the T5 model is the best-performing model, both with and without prompting. The mT5 model has the most parameters and is the worst without prompting but second-best with prompting. Conversely, the BART model has the smallest number of parameters and is the worst-performing model with prompting, while the mBART model is third-best with prompting and second-best without prompting.

The results for aspect category detection (slot 1) are the best, which is expected because aspect categories come from a fixed set of words. The aspect term extraction task (slot 2) is more challenging than the ACD task because aspect terms can be composed of varying numbers of words. Slot 1&2 is even more difficult as the models must simultaneously predict both the aspect term and category. The worst results are achieved for the TASD task because the models must correctly predict the polarity in addition to the aspect term and category.

6.8.1.2 Monolingual Results

Table 6.8 presents the monolingual results of the multilingual sequence-to-sequence models on datasets from various languages.

	Model							
	mT5				mBART			
Lang	Slot 1	Slot 2	Slot 1&2	TASD	Slot 1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning								
cs	$75.48^{\pm1.79}$	$66.49^{\pm2.46}$	$56.43^{\pm0.99}$	$47.98^{\pm0.99}$	$78.66^{\pm1.56}$	$78.91^{\pm1.31}$	$67.15^{\pm1.40}$	$57.48^{\pm1.71}$
en	$77.04^{\pm1.85}$	$69.20^{\pm2.17}$	$59.81^{\pm2.32}$	$53.38^{\pm2.49}$	$80.64^{\pm1.68}$	$80.82^{\pm0.94}$	$70.15^{\pm1.91}$	$62.82^{\pm1.95}$
es	$77.27^{\pm2.09}$	$73.10^{\pm2.09}$	$62.14^{\pm1.86}$	$56.17^{\pm2.41}$	$79.08^{\pm1.91}$	$76.64^{\pm1.16}$	$64.90^{\pm1.63}$	$59.91^{\pm1.99}$
fr	$72.27^{\pm2.51}$	$66.87^{\pm3.42}$	$52.79^{\pm3.63}$	$45.26^{\pm4.38}$	$72.59^{\pm2.23}$	$73.12^{\pm1.93}$	$57.53^{\pm2.39}$	$49.12^{\pm2.52}$
nl	$71.72^{\pm2.60}$	$65.90^{\pm3.13}$	$53.57^{\pm2.64}$	$46.68^{\pm2.22}$	$76.01^{\pm1.57}$	$73.96^{\pm2.95}$	$61.41^{\pm2.22}$	$54.25^{\pm2.73}$
ru	$82.17^{\pm2.91}$	$72.82^{\pm4.53}$	$65.92^{\pm4.69}$	$57.95^{\pm5.41}$	$83.22^{\pm1.90}$	$75.64^{\pm0.94}$	$68.41^{\pm1.89}$	$60.86^{\pm2.02}$
tr	$68.27^{\pm2.30}$	$46.28^{\pm3.35}$	$32.91^{\pm4.55}$	$26.85^{\pm4.11}$	$75.50^{\pm2.09}$	$62.32^{\pm3.28}$	$49.81^{\pm4.31}$	$39.63^{\pm4.68}$
Prompt-based fine-tuning								
cs	$\mathbf{85.45}^{\pm1.23}$	$\mathbf{84.80}^{\pm1.58}$	$\mathbf{74.99}^{\pm1.95}$	$\mathbf{67.30}^{\pm1.70}$	$83.25^{\pm0.71}$	$\mathbf{83.38}^{\pm0.62}$	$\mathbf{71.93}^{\pm1.60}$	$61.70^{\pm0.70}$
en	$\mathbf{87.20}^{\pm0.66}$	$\mathbf{86.85}^{\pm0.76}$	$\mathbf{79.12}^{\pm0.57}$	$\mathbf{73.59}^{\pm0.47}$	$84.56^{\pm0.74}$	$84.88^{\pm1.15}$	$75.34^{\pm0.98}$	$69.82^{\pm1.21}$
es	$\mathbf{85.53}^{\pm0.75}$	$\mathbf{80.66}^{\pm1.17}$	$\mathbf{72.38}^{\pm1.62}$	$\mathbf{68.03}^{\pm1.51}$	$82.46^{\pm0.50}$	$\mathbf{80.24}^{\pm0.84}$	$69.17^{\pm0.81}$	$63.78^{\pm0.44}$
fr	$\mathbf{80.77}^{\pm0.96}$	$\mathbf{80.38}^{\pm1.58}$	$\mathbf{67.70}^{\pm0.95}$	$\mathbf{60.56}^{\pm0.91}$	$72.69^{\pm1.50}$	$74.60^{\pm1.87}$	$57.77^{\pm3.29}$	$49.38^{\pm3.23}$
nl	$\mathbf{81.43}^{\pm1.28}$	$\mathbf{81.16}^{\pm1.82}$	$\mathbf{70.33}^{\pm1.10}$	$\mathbf{64.17}^{\pm0.91}$	$78.03^{\pm1.22}$	$75.31^{\pm2.32}$	$63.35^{\pm1.47}$	$55.46^{\pm1.62}$
ru	$\mathbf{86.79}^{\pm0.54}$	$\mathbf{81.88}^{\pm0.43}$	$\mathbf{75.00}^{\pm0.71}$	$\mathbf{68.61}^{\pm0.29}$	$85.11^{\pm0.93}$	$80.13^{\pm0.39}$	$72.02^{\pm1.10}$	$63.76^{\pm0.50}$
tr	$\mathbf{83.31}^{\pm1.76}$	$\mathbf{71.95}^{\pm0.83}$	$\mathbf{60.92}^{\pm0.77}$	$\mathbf{54.40}^{\pm1.92}$	$76.16^{\pm4.09}$	$62.09^{\pm3.39}$	$49.80^{\pm3.43}$	$39.17^{\pm3.20}$
State-of-the-art (Pontiki et al., 2016; Zhang, Deng et al., 2021)								
en	73.03	72.34	52.61	71.97	73.03	72.34	52.61	71.97
es	70.59	68.52	41.22	-	70.59	68.52	41.22	-
fr	61.21	66.67	47.72	-	61.21	66.67	47.72	-
nl	60.15	56.99	45.17	-	60.15	56.99	45.17	-
ru	64.83	49.31	39.44	-	64.83	49.31	39.44	-
tr	61.03	41.86	28.15	-	61.03	41.86	28.15	-

Table 6.8: Monolingual micro F1 scores (in %) of the mT5 and mBART models on different languages with prompt-based and traditional fine-tuning compared to state-of-the-art results if available. The best result (or results in case of overlapping confidence intervals) for each task and language is highlighted in **bold**.

The results indicate that prompt-based fine-tuning generally produces better results than traditional fine-tuning, with the mT5 model performing particularly well with prompting. While some results with the mBART model are better without prompting, most are improved with prompting. On the other hand, the mT5 model, without prompting, suffers from the same issues described earlier in the context of the English dataset. The worst results were obtained on the Turkish dataset, possibly due to its small size. All models outperformed the previous best results on all datasets

for the tasks described in (Pontiki et al., 2016) (slot 1, slot 2 and slot 1&2), which were achieved with simpler models. A literature review found no results for the TASD task other than English, which our method outperforms. Nevertheless, the best results are usually achieved on the English dataset, possibly because the models are pre-trained on more English data than data from other languages, leading to a better understanding of it.

6.8.1.3 Cross-lingual Results

Table 6.9 shows the zero-shot cross-lingual results (the models do not have annotated data from the target language available during fine-tuning) of multilingual models on different combinations of source and target languages.

As before, prompting performs better than traditional fine-tuning in most cases, especially with the mT5 model, which achieves the best results overall with prompting. The mT5 model with prompting performs best for every language combination and task, with the mBERT model with prompting achieving comparable results on a few tasks and language combinations. However, with traditional fine-tuning, the mBERT model outperforms the mT5 model in most cases.

Interestingly, when we compare the results of different language pairs, we find that results are usually better when the target language is English than when the source language is English. The reason may be that the pre-training data used for the models contained more English data than other languages, giving the models a better understanding of English. Therefore, fine-tuning specific tasks is more effective when other languages are used as source languages rather than target languages. The standard deviation and the confidence interval are larger for cross-results than for the monolingual results.

In terms of performance, the models achieve better results for the aspect category detection task (slot 1) than the aspect term extraction task (slot 2). This difference in performance could be attributed to the fact that aspect categories come from a fixed set of words, whereas aspect terms can have different words and lengths. Although the models can often identify individual words of the aspect term, they cannot always match the annotations exactly, leading to incorrect predictions. The difficulty of transferring knowledge from the source language to the target language for the aspect term detection task highlights that even the same phrase can vary considerably (e.g. in length) across languages. In monolingual experiments, the difference between slot 1 and slot 2 is significantly smaller, confirming the previous assumption. The results achieved with the mT5 model with prompting are often within 5% of the monolingual results for the aspect category detection task.

The models are selected based on their performance on the TASD task, so the results on the other tasks could be better. Although the models predicted all senti-

		Model							
		mT5				mBART			
SL	**TL**	Slot 1	Slot 2	Slot 1&2	TASD	Slo1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning									
en	cs	$70.36^{\pm 2.13}$	$35.06^{\pm 10.94}$	$32.74^{\pm 7.63}$	$27.75^{\pm 7.17}$	$63.00^{\pm 4.06}$	$52.83^{\pm 12.87}$	$39.78^{\pm 9.45}$	$31.35^{\pm 7.19}$
en	es	$68.50^{\pm 3.07}$	$33.64^{\pm 4.84}$	$32.68^{\pm 3.60}$	$28.55^{\pm 2.58}$	$53.36^{\pm 7.31}$	$50.24^{\pm 10.76}$	$35.76^{\pm 3.92}$	$28.91^{\pm 2.56}$
en	fr	$68.32^{\pm 1.51}$	$43.49^{\pm 5.70}$	$34.92^{\pm 3.83}$	$28.62^{\pm 3.54}$	$65.05^{\pm 3.55}$	$62.71^{\pm 1.82}$	$46.05^{\pm 2.34}$	$35.84^{\pm 1.98}$
en	nl	$59.47^{\pm 1.51}$	$31.84^{\pm 1.73}$	$27.61^{\pm 0.67}$	$23.50^{\pm 0.90}$	$66.93^{\pm 2.57}$	$53.88^{\pm 1.98}$	$44.13^{\pm 1.11}$	$37.12^{\pm 1.16}$
en	ru	$70.36^{\pm 5.17}$	$21.53^{\pm 12.45}$	$24.62^{\pm 7.75}$	$21.11^{\pm 6.81}$	$71.22^{\pm 4.94}$	$48.97^{\pm 7.39}$	$39.39^{\pm 6.95}$	$33.88^{\pm 5.88}$
en	tr	$58.22^{\pm 4.45}$	$19.87^{\pm 1.93}$	$12.03^{\pm 1.79}$	$10.21^{\pm 0.70}$	$60.53^{\pm 4.49}$	$37.16^{\pm 3.30}$	$24.50^{\pm 3.14}$	$20.43^{\pm 3.83}$
cs	en	$58.24^{\pm 4.41}$	$34.00^{\pm 1.64}$	$31.54^{\pm 1.21}$	$27.79^{\pm 0.79}$	$69.28^{\pm 3.63}$	$59.90^{\pm 6.76}$	$50.43^{\pm 5.21}$	$43.96^{\pm 5.01}$
es	en	$61.95^{\pm 1.47}$	$31.94^{\pm 4.00}$	$31.30^{\pm 3.06}$	$27.21^{\pm 3.17}$	$74.60^{\pm 1.60}$	$64.77^{\pm 3.54}$	$55.07^{\pm 4.64}$	$49.17^{\pm 3.96}$
fr	en	$68.25^{\pm 2.31}$	$40.89^{\pm 3.44}$	$38.12^{\pm 3.33}$	$32.72^{\pm 3.21}$	$76.20^{\pm 1.42}$	$70.79^{\pm 1.38}$	$58.99^{\pm 1.38}$	$51.88^{\pm 2.07}$
nl	en	$63.85^{\pm 3.42}$	$39.08^{\pm 4.60}$	$31.95^{\pm 2.14}$	$27.06^{\pm 1.51}$	$74.00^{\pm 2.66}$	$58.15^{\pm 4.88}$	$50.16^{\pm 4.90}$	$43.12^{\pm 5.18}$
ru	en	$57.60^{\pm 4.10}$	$26.76^{\pm 4.58}$	$31.85^{\pm 4.55}$	$26.42^{\pm 5.23}$	$66.74^{\pm 5.20}$	$59.59^{\pm 6.68}$	$51.11^{\pm 4.70}$	$46.02^{\pm 3.86}$
tr	en	$41.53^{\pm 2.49}$	$15.63^{\pm 2.97}$	$15.89^{\pm 2.50}$	$13.31^{\pm 2.45}$	$64.50^{\pm 2.68}$	$44.09^{\pm 2.47}$	$35.95^{\pm 2.25}$	$30.74^{\pm 1.49}$
Prompt-based fine-tuning									
en	cs	$\mathbf{80.21}^{\pm 0.85}$	$\mathbf{69.20}^{\pm 2.55}$	$\mathbf{59.32}^{\pm 1.96}$	$\mathbf{52.57}^{\pm 1.21}$	$64.83^{\pm 5.03}$	$53.87^{\pm 8.11}$	$38.06^{\pm 5.49}$	$28.90^{\pm 3.53}$
en	es	$\mathbf{78.60}^{\pm 1.03}$	$\mathbf{71.95}^{\pm 1.97}$	$\mathbf{60.65}^{\pm 1.00}$	$\mathbf{56.61}^{\pm 1.03}$	$66.81^{\pm 4.08}$	$65.13^{\pm 2.65}$	$49.15^{\pm 2.42}$	$42.36^{\pm 2.49}$
en	fr	$\mathbf{78.19}^{\pm 0.84}$	$\mathbf{74.44}^{\pm 1.36}$	$\mathbf{61.52}^{\pm 0.97}$	$\mathbf{53.21}^{\pm 0.98}$	$66.61^{\pm 1.79}$	$61.93^{\pm 2.40}$	$44.55^{\pm 2.08}$	$34.38^{\pm 2.28}$
en	nl	$\mathbf{78.37}^{\pm 1.01}$	$\mathbf{64.94}^{\pm 2.50}$	$\mathbf{59.13}^{\pm 1.07}$	$\mathbf{54.70}^{\pm 1.21}$	$68.42^{\pm 1.57}$	$54.90^{\pm 3.16}$	$45.69^{\pm 3.58}$	$38.32^{\pm 2.73}$
en	ru	$\mathbf{83.26}^{\pm 0.80}$	$\mathbf{66.98}^{\pm 1.60}$	$\mathbf{61.16}^{\pm 0.94}$	$\mathbf{55.22}^{\pm 1.54}$	$75.18^{\pm 2.30}$	$\mathbf{62.18}^{\pm 4.77}$	$50.95^{\pm 3.91}$	$43.88^{\pm 3.71}$
en	tr	$\mathbf{82.15}^{\pm 1.85}$	$\mathbf{57.50}^{\pm 3.27}$	$\mathbf{46.38}^{\pm 2.59}$	$\mathbf{42.86}^{\pm 2.45}$	$58.99^{\pm 8.56}$	$35.05^{\pm 3.30}$	$21.69^{\pm 4.24}$	$17.43^{\pm 3.99}$
cs	en	$\mathbf{79.46}^{\pm 0.82}$	$\mathbf{73.80}^{\pm 1.23}$	$\mathbf{63.34}^{\pm 1.02}$	$\mathbf{58.22}^{\pm 0.99}$	$75.42^{\pm 2.00}$	$\mathbf{70.79}^{\pm 2.46}$	$58.32^{\pm 2.12}$	$49.48^{\pm 1.01}$
es	en	$\mathbf{80.76}^{\pm 0.82}$	$\mathbf{72.50}^{\pm 1.80}$	$\mathbf{62.70}^{\pm 1.16}$	$\mathbf{58.60}^{\pm 1.26}$	$75.36^{\pm 1.70}$	$66.98^{\pm 3.52}$	$55.48^{\pm 2.28}$	$48.88^{\pm 2.00}$
fr	en	$\mathbf{84.73}^{\pm 0.62}$	$\mathbf{77.70}^{\pm 0.87}$	$\mathbf{70.02}^{\pm 1.01}$	$\mathbf{64.95}^{\pm 0.54}$	$78.54^{\pm 1.14}$	$\mathbf{75.41}^{\pm 1.60}$	$62.48^{\pm 0.47}$	$55.13^{\pm 0.41}$
nl	en	$\mathbf{79.67}^{\pm 1.22}$	$\mathbf{67.14}^{\pm 2.00}$	$\mathbf{59.28}^{\pm 1.45}$	$\mathbf{53.86}^{\pm 1.83}$	$\mathbf{78.84}^{\pm 1.08}$	$\mathbf{65.41}^{\pm 1.38}$	$\mathbf{57.44}^{\pm 1.04}$	$50.32^{\pm 0.93}$
ru	en	$\mathbf{81.00}^{\pm 0.52}$	$\mathbf{76.40}^{\pm 4.09}$	$\mathbf{67.10}^{\pm 3.18}$	$\mathbf{62.50}^{\pm 2.85}$	$76.92^{\pm 0.34}$	$\mathbf{78.73}^{\pm 0.84}$	$\mathbf{65.32}^{\pm 0.83}$	$59.04^{\pm 0.47}$
tr	en	$\mathbf{77.33}^{\pm 1.30}$	$\mathbf{64.46}^{\pm 4.29}$	$\mathbf{52.87}^{\pm 2.74}$	$\mathbf{48.02}^{\pm 2.89}$	$73.60^{\pm 1.60}$	$\mathbf{64.82}^{\pm 2.83}$	$\mathbf{51.29}^{\pm 2.16}$	$\mathbf{44.39}^{\pm 1.93}$

Table 6.9: Cross-lingual micro F1 scores (in %) of the mT5 and mBART models with traditional and prompt-based fine-tuning with different combinations of source and target languages (SL and TL, respectively). The best result (or results in case of overlapping confidence intervals) for each task and language combination is highlighted in **bold**.

ment triplets, for slot 1 and slot 2, only one part is extracted, and for slot 1&2, only two parts are extracted. We believe the results of these tasks could be improved if the model explicitly focused on them.

Another challenge the used models face is that they must produce the output exactly in our given format. If the models make a mistake, we cannot correctly parse the output, leading to a drop in performance. For instance, if the model generates the sentence *"Food quality was great, given the expression: steak"* but uses *"was"* instead of *"is"*, the entire sentence is discarded, despite correctly predicting the aspect category, sentiment polarity, and aspect term. Moreover, the annotation process for datasets

may vary slightly for each language, which could be another potential issue. Lastly, all outputs are in English because the annotations are in English, which might explain why the results are better with English as the target language. Combining multiple languages could also cause difficulties for the models.

Table 6.10 compares the cross-lingual and monolingual results on the TASD task. The comparison on this task is presented because the sequence-to-sequence models are primarily fine-tuned on the TASD task. The mT5 model with traditional fine-tuning generally achieves the worst cross-lingual results compared to the monolingual results, with the most significant difference being 40%. The results achieved with both fine-tuning styles for the mBART model and prompting for the mT5 model are usually about 10 to 20% worse than the monolingual results. Since these models did not see any target language data during fine-tuning, some results can be considered good.

6.8.2 Sentiment Polarity Classification Results

This subsection presents the results of the models for sentiment polarity classification that solve slot 3 in the task described in (Pontiki et al., 2016) task, i.e. classify the polarity of a given aspect term and aspect category pair.

6.8.2.1 Results on the English Dataset

Table 6.11 shows the results of five models on the English dataset.

Traditional fine-tuning and prompt-based fine-tuning show no significant difference in most cases. However, the ELECTRA model performs better with traditional fine-tuning by over 3%, possibly due to the discriminator part being used for traditional fine-tuning. For prompting, only the generator part is trained for masked language modelling. Additionally, the ELECTRA model used with prompting has more than six times fewer parameters than the model used without prompting. In most cases, the RoBERTa, XLM-R and ELECTRA models outperform the BERT and mBERT models. These models have modified BERT architecture, which should improve their performance. On the other hand, the mBERT model has fewer parameters and an older architecture, which may account for its poorer performance. Furthermore, mBERT is the only model not significantly surpassing the previous state-of-the-art result achieved with a simpler model.

6.8.2.2 Monolingual Results

Table 6.12 shows the monolingual results achieved with sentiment polarity classification models and compares them to state-of-the-art results.

		mT5		mBART	
Source	**Target**	TR-FT	PT-FT	TR-FT	PT-FT
en	cs	$27.75^{\pm 7.17}$	$52.57^{\pm 1.21}$	$31.35^{\pm 7.19}$	$28.90^{\pm 3.53}$
cs (mono)		$47.98^{\pm 0.99}$	$67.30^{\pm 1.70}$	$57.48^{\pm 1.71}$	$61.70^{\pm 0.70}$
en	es	$28.55^{\pm 2.58}$	$56.61^{\pm 1.03}$	$28.91^{\pm 2.56}$	$42.36^{\pm 2.49}$
es (mono)		$56.17^{\pm 2.41}$	$68.03^{\pm 1.51}$	$59.91^{\pm 1.99}$	$63.78^{\pm 0.44}$
en	fr	$28.62^{\pm 3.54}$	$53.21^{\pm 0.98}$	$35.84^{\pm 1.98}$	$34.38^{\pm 2.28}$
fr (mono)		$45.26^{\pm 4.38}$	$60.56^{\pm 0.91}$	$49.12^{\pm 2.52}$	$49.38^{\pm 3.23}$
en	nl	$23.50^{\pm 0.90}$	$54.70^{\pm 1.21}$	$37.12^{\pm 1.16}$	$38.32^{\pm 2.73}$
nl (mono)		$46.68^{\pm 2.22}$	$64.17^{\pm 0.91}$	$54.25^{\pm 2.73}$	$55.46^{\pm 1.62}$
en	ru	$21.11^{\pm 6.81}$	$55.22^{\pm 1.54}$	$33.88^{\pm 5.88}$	$43.88^{\pm 3.71}$
ru (mono)		$57.95^{\pm 5.41}$	$68.61^{\pm 0.29}$	$60.86^{\pm 2.02}$	$63.76^{\pm 0.50}$
en	tr	$10.21^{\pm 0.70}$	$42.86^{\pm 2.45}$	$20.43^{\pm 3.83}$	$17.43^{\pm 3.99}$
tr (mono)		$26.85^{\pm 4.11}$	$54.40^{\pm 1.92}$	$39.63^{\pm 4.68}$	$39.17^{\pm 3.20}$
cs	en	$27.79^{\pm 0.79}$	$58.22^{\pm 0.99}$	$43.96^{\pm 5.01}$	$49.48^{\pm 1.01}$
es	en	$27.21^{\pm 3.17}$	$58.60^{\pm 1.26}$	$49.17^{\pm 3.96}$	$48.88^{\pm 2.00}$
fr	en	$32.72^{\pm 3.21}$	$64.95^{\pm 0.54}$	$51.88^{\pm 2.07}$	$55.13^{\pm 0.41}$
nl	en	$27.06^{\pm 1.51}$	$53.86^{\pm 1.83}$	$43.12^{\pm 5.18}$	$50.32^{\pm 0.93}$
ru	en	$26.42^{\pm 5.23}$	$62.50^{\pm 2.85}$	$46.02^{\pm 3.86}$	$59.04^{\pm 0.47}$
tr	en	$13.31^{\pm 2.45}$	$48.02^{\pm 2.89}$	$30.74^{\pm 1.49}$	$44.39^{\pm 1.93}$
en (mono)		$53.38^{\pm 2.49}$	$73.59^{\pm 0.47}$	$62.82^{\pm 1.95}$	$69.82^{\pm 1.21}$

Table 6.10: Cross-lingual micro F1 scores (in %) of the mT5 and mBART models with traditional and prompt-based fine-tuning compared to monolingual results on the TASD task.

Once again, the comparison shows that, in most cases, there is generally no significant difference between traditional and prompt-based fine-tuning. However, using prompting gives both models some advantage on the Dutch dataset. Across all languages, the XLM-R model consistently outperforms the mBERT model, likely due to its higher number of parameters and more advanced pre-training approach. Notably, the XLM-R model significantly outperforms the previous state-of-the-art models, in some cases by more than 10%. The mBERT model also surpasses the previous SOTA models but to a lesser extent. It is worth noting that the previous state-of-the-art results were obtained with simpler models with fewer parameters.

Model	TR-FT	PT-FT	SOTA
BERT	**93.91**$^{\pm 0.34}$	92.55$^{\pm 0.88}$	
ELECTRA	**95.04**$^{\pm 1.29}$	91.48$^{\pm 0.70}$	
RoBERTa	95.30$^{\pm 0.26}$	95.42$^{\pm 0.54}$	88.13
mBERT	87.39$^{\pm 0.58}$	**88.47**$^{\pm 0.35}$	
XLM-R	95.01$^{\pm 0.47}$	94.77$^{\pm 0.34}$	

Table 6.11: Accuracy scores (in %) for the English dataset and different sentiment polarity classification models with prompt-based fine-tuning (PT-FT) and traditional fine-tuning (TR-FT), along with the state-of-the-art results (SOTA) described in (Pontiki et al., 2016). Results in **bold** indicate significantly better performance between the two fine-tuning styles for a given model.

	mBERT		XLM-R		SOTA
Lang	TR-FT	PT-FT	TR-FT	PT-FT	
cs	77.42$^{\pm 1.71}$	**80.31**$^{\pm 1.07}$	88.59$^{\pm 1.02}$	88.40$^{\pm 1.48}$	-
en	87.39$^{\pm 0.58}$	**88.47**$^{\pm 0.35}$	95.01$^{\pm 0.47}$	94.77$^{\pm 0.34}$	88.13
es	88.80$^{\pm 0.60}$	87.97$^{\pm 0.35}$	93.72$^{\pm 0.58}$	94.29$^{\pm 0.58}$	83.58
fr	81.82$^{\pm 1.00}$	81.89$^{\pm 0.49}$	90.31$^{\pm 0.85}$	90.31$^{\pm 0.30}$	78.83
nl	78.43$^{\pm 2.61}$	**83.36**$^{\pm 0.80}$	88.93$^{\pm 0.94}$	**91.55**$^{\pm 0.56}$	77.81
ru	84.43$^{\pm 0.42}$	82.95$^{\pm 1.35}$	91.30$^{\pm 0.87}$	90.70$^{\pm 0.39}$	77.92
tr	**85.35**$^{\pm 1.84}$	78.99$^{\pm 2.25}$	93.78$^{\pm 0.89}$	93.33$^{\pm 1.18}$	84.28

Table 6.12: Monolingual accuracy scores (in %) for mBERT and XLM-R models with traditional fine-tuning (TR-FT) and prompt-based fine-tuning (PT-FT), along with the state-of-the-art results (SOTA) for a given language described in (Pontiki et al., 2016). Results in **bold** indicate significantly better performance between the two fine-tuning styles for a given model.

6.8.2.3 Cross-lingual Results

Table 6.13 shows the results of cross-lingual experiments and compares them with monolingual results.

Similar to the monolingual results, prompt-based and traditional fine-tuning show similar performance in most cases. The XLM-R model consistently outperforms the mBERT model by 5 to 20%, usually around 10%. Additionally, the results achieved by the XLM-R model are closer to monolingual results. In some cases, the difference is less than 1%, which is an excellent result given that the model did not

		mBERT		XLM-R	
Source	**Target**	TR-FT	PT-FT	TR-FT	PT-FT
en	cs	$71.99^{\pm 2.86}$	$74.20^{\pm 0.69}$	$89.99^{\pm 0.59}$	$89.02^{\pm 0.46}$
cs (mono)		$77.42^{\pm 1.71}$	$80.31^{\pm 1.07}$	$88.59^{\pm 1.02}$	$88.40^{\pm 1.48}$
en	es	$82.06^{\pm 0.62}$	$82.11^{\pm 0.59}$	$92.52^{\pm 0.26}$	$93.00^{\pm 0.34}$
es (mono)		$88.80^{\pm 0.60}$	$87.97^{\pm 0.35}$	$93.72^{\pm 0.58}$	$94.29^{\pm 0.58}$
en	fr	$67.66^{\pm 2.56}$	$69.92^{\pm 1.77}$	$\mathbf{86.67}^{\pm 0.90}$	$84.72^{\pm 0.88}$
fr (mono)		$81.82^{\pm 1.00}$	$81.89^{\pm 0.49}$	$90.31^{\pm 0.85}$	$90.31^{\pm 0.30}$
en	nl	$72.27^{\pm 0.84}$	$73.21^{\pm 0.84}$	$90.32^{\pm 0.66}$	$91.32^{\pm 0.54}$
nl (mono)		$78.43^{\pm 2.61}$	$83.36^{\pm 0.80}$	$88.93^{\pm 0.94}$	$91.55^{\pm 0.56}$
en	ru	$77.31^{\pm 1.83}$	$75.07^{\pm 0.84}$	$90.44^{\pm 0.66}$	$89.95^{\pm 0.23}$
ru (mono)		$84.43^{\pm 0.42}$	$82.95^{\pm 1.35}$	$91.30^{\pm 0.87}$	$90.70^{\pm 0.39}$
en	tr	$\mathbf{77.71}^{\pm 1.97}$	$73.08^{\pm 1.40}$	$93.43^{\pm 1.75}$	$92.08^{\pm 0.89}$
tr (mono)		$85.35^{\pm 1.84}$	$78.99^{\pm 2.25}$	$93.78^{\pm 0.89}$	$93.33^{\pm 1.18}$
cs	en	$72.73^{\pm 2.97}$	$77.69^{\pm 2.07}$	$91.06^{\pm 1.23}$	$90.29^{\pm 0.48}$
es	en	$78.82^{\pm 1.58}$	$79.53^{\pm 1.14}$	$\mathbf{93.08}^{\pm 0.15}$	$92.08^{\pm 0.34}$
fr	en	$77.65^{\pm 1.15}$	$75.95^{\pm 1.67}$	$\mathbf{93.91}^{\pm 0.91}$	$91.59^{\pm 0.45}$
nl	en	$80.23^{\pm 1.68}$	$80.00^{\pm 1.09}$	$92.77^{\pm 0.49}$	$92.36^{\pm 0.30}$
ru	en	$79.28^{\pm 0.47}$	$79.49^{\pm 0.49}$	$93.60^{\pm 0.20}$	$92.99^{\pm 0.52}$
tr	en	$79.28^{\pm 0.47}$	$77.69^{\pm 1.19}$	$\mathbf{92.30}^{\pm 0.57}$	$90.62^{\pm 0.57}$
en (mono)		$87.39^{\pm 0.58}$	$88.47^{\pm 0.35}$	$95.01^{\pm 0.47}$	$94.77^{\pm 0.34}$

Table 6.13: Cross-lingual accuracy scores (in %) for mBERT and XLM-R models with traditional fine-tuning (TR-FT) and prompt-based fine-tuning (PT-FT), along with their respective monolingual results for comparison. Results in **bold** indicate significantly better performance between the two fine-tuning styles for a given model and language combination.

have access to data in the target language during fine-tuning and is fine-tuned only on data from the source language.

The XLM-R model shows the most significant difference of 6% with prompting, with English as the source language and French as the target language, when comparing the cross-lingual results with monolingual results on the target language. However, when traditional fine-tuning is used, the XLM-R model performs better with English as the source language and Dutch or Czech as the target language than monolingual results in the corresponding target language. On the other hand, cross-lingual results achieved by the mBERT model are worse than the monolingual

results by around 8% in most cases, with the largest difference being 14% and the smallest 5%. As with the XLM-R model, the most significant difference is observed when the source language is English and the target language is French. This difference may indicate that the French test dataset is most dissimilar to the English training data.

6.8.3 Evaluation of Results

Our proposed sequence-to-sequence method is an efficient solution for aspect-based sentiment analysis in different languages, achieving several new state-of-the-art results on the used datasets. In most cases, using prompting with sequence-to-sequence models outperforms traditional fine-tuning. The mT5 model benefits the most from using prompting. It may be because the model has some prior information about the number of sentiment triplets it should generate, which the traditional fine-tuned model does not have. Another reason might be that the prompting is better for the sequence-to-sequence models because it closely matches their pre-training objectives.

The target-aspect-sentiment detection (TASD) task is the most challenging task solved in this thesis because the model must simultaneously predict all the aspect categories, aspect terms and sentiment polarities. The aspect category detection task is the least difficult, especially in cross-lingual settings where the models have difficulty identifying aspect terms correctly for the aspect term extraction task.

In the case of cross-lingual aspect-based sentiment analysis, the task is difficult when used in a zero-shot cross-lingual setting, i.e. the model is not fine-tuned on any data of the target language. The models perform well on the aspect category detection task, where the results are often within 5% of the monolingual results. The aspect term extraction task is more complex than the aspect category detection task because the models often predict only part of the aspect term and not the exact term, possibly because of language differences. The TASD task is the most difficult, and the results are often worse by more than 10% compared to monolingual results.

In the case of sentiment polarity classification, there is no significant difference between prompt-based and traditional fine-tuning in most cases. The proposed solution achieves state-of-the-art results on the used datasets. Cross-lingual experiments often yield results within a few per cent of the monolingual results. In some cases, the cross-lingual results even outperform the monolingual results.

Overall, the zero-shot cross-lingual aspect-based sentiment analysis proves to be a difficult task, and better results are often achieved with models with more parameters.

6.9 Attempts to Improve the Performance

We experimented with various settings to improve the performance of our models and combined them in different ways.

We attempted to improve the performance by modifying the mapping function for all our models. One such modification was adding a special <null> token to the tokenizer vocabulary and using it when the aspect term was "NULL". However, this did not significantly change the results, so we decided to use the original formulation without modifying the tokenizer vocabulary. We also tried closing the attribute type in parentheses (e.g. *"(quality)"* instead of *"quality"*), but this had no significant impact on the results. We decided to omit the parentheses to reduce the size of the text.

For the sequence-to-sequence models (both with and without prompting), we tried using the <sep> token instead of *";"* to concatenated transformed triplets. For the T5 and mT5 models, we had to add the special token into the tokenizer vocabulary. The BART and mBART models already have the separation token in the tokenizer vocabulary, but it is the same as the token used to denote the end of the sequence. As a result, these models only returned up to one transformed triplet. We tried redefining the token to solve this issue, but it yielded worse results than using *";"*. Therefore, we decided to stick with the semicolon. We also experimented with entirely different outputs in the form of *"aspect: <aspect>, category: <category>, polarity: <polarity>"*, again without any significant changes in the results. Last, we tried using a special <empty> token for the original experiments, including examples without triplets. However, the results were not better than those using an empty string.

For the models for sentiment polarity classification using prompting, we experimented with two variants of optimizing the cross-entropy loss during training. The first variant calculated the loss over all tokens from the tokenizer vocabulary. The other variant calculated the loss only over the tokens (words) used for the mapping of sentiment polarity (specifically *"great"*, *"ok"*, and *"bad"*). In most cases, the first variant outperformed the second variant by more than 10%, so we decided to use the first option. We also conducted a few experiments where we calculated the loss and only made predictions for the words used for mapping sentiment polarity. In this case, we considered only the most probable mapping words. However, we found no improvement over the original approach of computing the loss and making predictions over all words in the vocabulary. Therefore, we decided to stick with the original version.

Conclusion 7

This thesis focuses on cross-lingual aspect-based sentiment analysis and explores different tasks in ABSA and possible solutions using Transformer-based models.

This thesis proposes a sequence-to-sequence method for solving various ABSA tasks simultaneously, including aspect term extraction and target-aspect-sentiment detection. Models using this method are evaluated on benchmark datasets in multiple languages. It establishes the baseline results for the new Czech dataset created for this thesis and achieves new state-of-the-art results on datasets from other languages. The method can be used with both prompt-based and traditional fine-tuning. Prompting is particularly effective, especially with the T5 model and its multilingual version, which achieve the best overall result with this method. The reason might be that prompting matches these models' pre-training objectives closely.

The method is also evaluated on zero-shot cross-lingual ABSA with different language combinations, where the model is not fine-tuned on any annotated data from the target language. It shows promising results, with the best models achieving micro F1 score within 5 to 15% of the monolingual results for some language combinations. The mT5 model with prompting is the best-performing model.

Additionally, the thesis proposes a method for classifying the sentiment polarity of aspect terms and categories with traditional and prompt-based fine-tuning. It achieves new state-of-the-art results in various languages and sets a new baseline for the Czech dataset. The cross-lingual results are also promising, often within 2% of the monolingual results, in some cases even better. In this case, there is usually no significant difference between prompt-based and traditional fine-tuning.

All the previous points correspond to the objectives of this thesis. Future work may focus on different combinations of source and target languages, examining the impact of adding different language sources to the training set and exploring prompt design and sentiment mapping. Furthermore, it is possible to perform a few-shot cross-lingual aspect-based sentiment analysis to determine the smallest number of examples required to achieve comparable results to monolingual experiments.

All experiments were performed on the MetaCentrum distributed computing infrastructure, involving more than 7500 experiments.

List of Abbreviations

ABSA	aspect-based sentiment analysis
ACD	aspect category detection
ACSA	aspect category sentiment analysis
ACSD	aspect-category-sentiment detection
ASQP	aspect sentiment quad prediction
AOCE	aspect opinion co-extraction
AOPE	aspect-opinion pair extraction
ASC	aspect sentiment classification
ASTE	aspect sentiment triplet extraction
ATE	aspect term extraction
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BPE	byte-pair encoding
CI	confidence interval
CNN	convolutional neural network
E2E-ABSA	end-to-end aspect-based sentiment analysis
ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
FN	false negative
FP	false positive

GPT	Generative Pre-trained Transformer
GRU	gated recurrent unit
IAA	inter-annotator agreement
LM	language model, language modelling
LSTM	long short-term memory
MLM	masked language model, masked language modelling
NLP	natural language processing
NSP	next sentence prediction
OOV	out-of-vocabulary
OTE	opinion terms extraction
PT-FT	prompt-based fine-tuning
RNN	recurrent neural network
RoBERTa	A Robustly Optimized BERT Pre-training Approach
RTD	replaced token detection
SA	sentiment analysis
SB1	first subtask in (Pontiki et al., 2016)
Seq2Seq	sequence-to-sequence
SL	source language
SOTA	state-of-the-art
T5	Text-To-Text Transfer Transformer
TABSA	targeted aspect-based sentiment analysis
TASD	target-aspect-sentiment detection
TL	target language
TN	true negatives
TOWE	target-oriented opinion word extraction

TP true positives

TR-FT traditional-based fine-tuning

UABSA unified aspect-based sentiment analysis

XLM cross-lingual language models

XLM-R XLM-RoBERTa

Bibliography

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. *Advances in neural information processing systems, 19*.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Barnes, J., Oberlaender, L., Troiano, E., Kutuzov, A., Buchmann, J., Agerri, R., Øvrelid, L., & Velldal, E. (2022). SemEval 2022 task 10: Structured sentiment analysis. *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, 1280–1295. https://doi.org/10.18653/v1/2022.semeval-1.180

Biewald, L. (2020). Experiment tracking with weights and biases [Software available from wandb.com]. https://www.wandb.com/

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33*, 1877–1901.

Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111. https://doi.org/10.3115/v1/W14-4012

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. *ICLR*. https://openreview.net/pdf?id=r1xMH1BtvB

Conneau, A., & Lample, G. (2019). Cross-lingual language model pretraining. In *Proceedings of the 33rd international conference on neural information processing systems*. Curran Associates Inc.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Detlefsen, N. S., Borovec, J., Schock, J., Jha, A. H., Koker, T., Liello, L. D., Stancl, D., Quan, C., Grechkin, M., & Falcon, W. (2022). Torchmetrics - measuring reproducibility in pytorch. *Journal of Open Source Software*, *7*(70), 4101. https://doi.org/10.21105/joss.04101

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *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)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Evgeniou, T., & Pontil, M. (2004). Regularized multi–task learning. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 109–117.

Falcon, W., & The PyTorch Lightning team. (2019). *PyTorch Lightning* (Version 1.4). https://doi.org/10.5281/zenodo.3828935

Freitag, M., & Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. *Proceedings of the First Workshop on Neural Machine Translation*, 56–60. https://doi.org/10.18653/v1/W17-3207

Gao, T., Fang, J., Liu, H., Liu, Z., Liu, C., Liu, P., Bao, Y., & Yan, W. (2022). LEGO-ABSA: A prompt-based task assemblable unified generative framework for multi-task aspect-based sentiment analysis. *Proceedings of the 29th International Conference on Computational Linguistics*, 7002–7012. https://aclanthology.org/2022.coling-1.610

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. *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)*, 3816–3830. https://doi.org/10.18653/v1/2021.acl-long.295

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., et al. (2021). Pre-trained models: Past, present and future. *AI Open*, *2*, 225–250.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hercig, T., Brychcı´n, T., Svoboda, L., Konkol, M., & Steinberger, J. (2016). Unsupervised methods to improve aspect-based sentiment analysis in czech. *Computación y Sistemas*, *20*(3), 365–375.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Indurkhya, N., & Damerau, F. J. (2010). *Handbook of natural language processing* (2nd). Chapman & Hall/CRC.

Jiang, Z., Xu, F. F., Araki, J., & Neubig, G. (2020). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, *8*, 423–438.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd edition)*. Prentice-Hall, Inc.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 66–75. https://doi.org/10.18653/v1/P18-1007

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. https://doi.org/10.18653/v1/D18-2012

Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E., & Hubbard, W. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, *27*(11), 41–46.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *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)*, 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European conference on computer vision*, 740–755.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, *5*(1), 1–167.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, *55*(9). https://doi.org/10.1145/3560815

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR, abs/1907.11692*. http://arxiv.org/abs/1907.11692

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., & Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, *8*, 726–742.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. https://openreview.net/forum?id=Bkg6RiCqY7

Manning, C. D., Schütze, H., & Raghavan, P. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Y. Bengio & Y. LeCun (Eds.), *1st international conference on learning representations, ICLR 2013, scottsdale, arizona, usa, may 2-4, 2013, workshop track proceedings*. http://arxiv.org/abs/1301.3781

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2227–2237. https://doi.org/10.18653/v1/N18-1202

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 486–495. https://doi.org/10.18653/v1/S15-2082

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 task 4: Aspect based sentiment analysis. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 27–35. https://doi.org/10.3115/v1/S14-2004

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., & Eryiğit, G. (2016). SemEval-2016 task 5: Aspect based sentiment analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 19–30. https://doi.org/10.18653/v1/S16-1002

Přibáň, P., Šmíd, J., Mištera, A., & Král, P. (2022). Linear transformations for cross-lingual sentiment analysis. In P. Sojka, A. Horák, I. Kopeček & K. Pala (Eds.), *Text, speech, and dialogue* (pp. 125–137). Springer International Publishing.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

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

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, *21*(140), 1–67.

Saeidi, M., Bouchard, G., Liakata, M., & Riedel, S. (2016). SentiHood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 1546–1556. https://aclanthology.org/C16-1146

Schuster, M., & Nakajima, K. (2012). Japanese and korean voice search. *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5149–5152.

Shazeer, N. (2020). GLU variants improve transformer. *CoRR, abs/2002.05202*. https://arxiv.org/abs/2002.05202

Shazeer, N., & Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning, ICML 2018, stockholmsmässan, stock-*

holm, sweden, july 10-15, 2018 (pp. 4603–4611). PMLR. http://proceedings.mlr.press/v80/shazeer18a.html

Sido, J., Pražák, O., Přibáň, P., Pašek, J., Seják, M., & Konopı´k, M. (2021). Czert – Czech BERT-like model for language representation. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 1326–1338. https://aclanthology.org/2021.ranlp-1.149

Straka, M., Náplava, J., Straková, J., & Samuel, D. (2021). Robeczech: Czech roberta, a monolingual contextualized language representation model. In K. Ekštein, F. Pártl & M. Konopík (Eds.), *Text, speech, and dialogue* (pp. 197–209). Springer International Publishing.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., Gu, J., & Fan, A. (2020). Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Thrun, S., & Pratt, L. (1998). Learning to learn: Introduction and overview. In *Learning to learn* (pp. 3–17). Springer.

Trinh, T. H., & Le, Q. V. (2018). A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., … Rush, A. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). MT5: A massively multilingual pre-trained text-to-text transformer. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 483–498. https://doi.org/10.18653/v1/2021.naacl-main.41

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, *10*, 291–306. https://doi.org/10.1162/tacl_a_00461

Zhang, W., Deng, Y., Li, X., Yuan, Y., Bing, L., & Lam, W. (2021). Aspect sentiment quad prediction as paraphrase generation. *Proceedings of the 2021 Conference*

on Empirical Methods in Natural Language Processing, 9209–9219. https://doi.org/10.18653/v1/2021.emnlp-main.726

Zhang, W., He, R., Peng, H., Bing, L., & Lam, W. (2021). Cross-lingual aspect-based sentiment analysis with aspect term code-switching. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 9220–9230. https://doi.org/10.18653/v1/2021.emnlp-main.727

Zhang, W., Li, X., Deng, Y., Bing, L., & Lam, W. (2021). Towards generative aspect-based sentiment analysis. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 504–510. https://doi.org/10.18653/v1/2021.acl-short.64

Zhang, W., Li, X., Deng, Y., Bing, L., & Lam, W. (2022). A survey on aspect-based sentiment analysis: Tasks, methods, and challenges. *IEEE Transactions on Knowledge and Data Engineering*.

Zhong, Z., Friedman, D., & Chen, D. (2021). Factual probing is [MASK]: Learning vs. learning to recall. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5017–5033. https://doi.org/10.18653/v1/2021.naacl-main.398

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE international conference on computer vision*, 19–27.

User Documentation

The attached ZIP file contains this thesis's source code, data and files.

A.1 Contents of the Attached ZIP

The attached ZIP contains the following folders and files:

- `Text_thesis` – Folder containing the text of this master's thesis, source LaTeX files and images.
  - `latex` – Folder containing all the necessary files to create the text of the thesis, e.g. LaTeX source files (individual chapters are in the `chapters` subfolder) and images (in the `img` subfolder).
  - `DP_Jakub_Smid.pdf` – Thesis in PDF format.
- `Poster` – Folder containing the poster created for presenting this master's thesis.
  - `Smid_Jakub_2023.pdf` – Poster in PDF format.
  - `Smid_Jakub_2023.pub` – Source file of the poster.
- `Application_and_libraries` – Folder containing the source code of the application.
  - `data` – Folder to which the data from the `Input_data` folder must be copied to run the application.
  - `src` – Folder containing the source code of the application.
  - `tests` – Folder containing the test data and source code for testing the application.
  - `main.py` – Main file of the application.
  - `README.md` – User manual for the application with a few examples of running the experiments.

  - `requirements.txt` – File with the list of required libraries.

- `Input_data` – Folder containing subfolders with all data needed to run the application. Each subfolder contains `train.xml` with training data and `test.xml` with test data. To run the application, the content of this folder must be copied to the `data` folder in the root of the application (in the `Application_and_libraries` folder).
  - `cs` – Folder containing the Czech data.
  - `en` – Folder containing the English data.
  - `es` – Folder containing the Spanish data.
  - `fr` – Folder containing the French data.
  - `nl` – Folder containing the Dutch data.
  - `ru` – Folder containing the Russian data.
  - `tr` – Folder containing the Turkish data.
- `Results` – Folder containing the results of the conducted experiments.
  - `results.xlsx` – File with the results of the best experiments performed, which are presented in the thesis text.
  - `results_wandb.csv` – File with the results of all conducted experiments exported from `WandB`.
- `Readme.txt` – File with the structure of the ZIP folder.

A.2 User Manual

This section describes the usage of the application.

1. Make sure to have Python 3.10 and pip installed.
2. Navigate to the `Application_and_libraries` folder.
3. To install requirements, run `pip install -r requirements.txt`.
4. Create a `data` folder and copy all the contents of the `Input_data` folder from the root folder of the ZIP file into it.
5. Run `python main.py` with the optional parameters listed below. A few examples for running the experiments are in the `README.md` file in the `Application_and_libraries` folder and in the `results.xlsx` file in the `Results` folder.

The program takes several optional parameters:

- `--model` – Name or path to pre-trained model (default is "t5-base").
- `--batch_size` – Batch size. The default value is 64. Note that this is the global batch size for multi-GPU training, not the per-GPU batch size (the number provided will be divided by the number of GPUs used).
- `--max_seq_length` – Maximum sequence length. The default value is 256.
- `--max_seq_length_label` – Maximum sequence length for labels. The default value is 512.
- `--lr` – Learning rate. The default value is 0.0001.
- `--epochs` – Number of training epochs. The default value is 10.
- `--test_language` – Language of test dataset (target language). The default value is "en". Options:
  - cs – Czech.
  - en – English.
  - es – Spanish.
  - fr – French.
  - nl – Dutch.
  - ru – Russian.
  - tr – Turkish.
- `--train_language` – Language of the training dataset (source language). The default value is "en". Options:
  - cs – Czech.
  - en – English.
  - es – Spanish.
  - fr – French.
  - nl – Dutch.
  - ru – Russian.
  - tr – Turkish.
- `--optimizer` – Optimizer for training. Options:
  - adafactor – AdaFactor. Default value.

  - AdamW – AdamW.

- `--mode` – Mode of the training. The default value is "dev". Options:
  - dev – Splits the training data into training and validation sets. The validation set is used for selecting the best model, which is then evaluated on the test set of the target language. In the case of different source and target languages, the validation set is taken from the training data of the target language and the whole training data is used for training.
  - test – Uses whole training dataset from the source language for training and test dataset from the target language for evaluation.
- `--checkpoint_monitor` – Metric based on which the best model will be stored according to the performance on validation data in "dev" mode. The default value is val_loss. Options:
  - val_loss – Validation loss.
  - slot1_2_f1 – F1 score on Slot 1&2.
  - slot1_f1 – F1 score on Slot 1.
  - slot2_f2 – F1 score on Slot 2.
  - slot3_acc – Accuracy score on Slot 3.
  - tasd_f1 – F1 score on TASD task.
- `--accumulate_grad_batches` – Accumulates gradient batches. The default value is 1. It is used when there is insufficient memory for training for the required effective batch size.
- `--beam_size` – Beam size for beam search decoding. The default value is 1.
- `--prompting` – Use prompting.
- `--classification` – Use classification.

A.2.1 Restrictions and Details

This subsection provides some restrictions and details to the application and its usage.

- `--classification` cannot be used for sequence-to-sequence models (e.g. T5).
- When selecting a model for sentiment polarity classification (e.g. BERT), either `--classification` or `--prompting` has to be used.

- The `--prompting` and `--classification` options cannot be selected together.
- The user is responsible for selecting the correct `--checkpoint_monitor` (e.g. `slot3_accuracy` is not measured when using sequence-to-sequence models).
- The `--model` argument containing the substring *"t5"* or *"bart"* indicates the use of a sequence-to-sequence model.
- The program automatically detects whether it is possible to use GPU for training and the number of available GPUs.
- The program tries to use *WandB* for logging the metrics. It uses logging into the CSV file if it is unavailable or not possible (e.g. the user is not logged in or does not have permission to access the project).

Additional Selected Results of Experiments

B

This appendix shows the results of the sequence-to-sequence models when the examples without any annotation sentiment triplets were not filtered for the training and evaluation. For these examples, the models are supposed to return an empty string. Table B.1 shows the results achieved on the English dataset. Table B.2 shows the monolingual results achieved with multilingual models on different languages. Table B.3 shows the cross-lingual results achieved with multilingual models on different combinations of source and target languages.

Model	Slot 1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning				
T5	$83.06^{\pm 0.78}$	$82.86^{\pm 0.95}$	$75.52^{\pm 0.37}$	$71.77^{\pm 0.32}$
mT5	$72.15^{\pm 2.04}$	$63.01^{\pm 4.04}$	$54.40^{\pm 3.76}$	$48.62^{\pm 4.08}$
BART	$74.80^{\pm 0.27}$	$69.05^{\pm 1.14}$	$63.96^{\pm 0.45}$	$60.51^{\pm 0.77}$
mBART	$76.95^{\pm 1.21}$	$76.88^{\pm 1.81}$	$66.87^{\pm 1.93}$	$60.63^{\pm 2.01}$
Prompt-based fine-tuning				
T5	$87.71^{\pm 0.53}$	$87.31^{\pm 1.22}$	$79.83^{\pm 0.58}$	$74.82^{\pm 0.47}$
mT5	$86.46^{\pm 0.90}$	$84.79^{\pm 0.76}$	$76.78^{\pm 0.62}$	$71.67^{\pm 0.57}$
BART	$79.03^{\pm 0.79}$	$59.86^{\pm 0.94}$	$58.26^{\pm 0.82}$	$53.85^{\pm 0.54}$
mBART	$84.62^{\pm 0.47}$	$85.69^{\pm 0.73}$	$75.82^{\pm 1.02}$	$69.75^{\pm 1.07}$

Table B.1: Micro F1 scores (in %) of the sequence-to-sequence models on the English dataset with traditional and prompt-based fine-tuning where the examples without any annotation triplets were not filtered from the dataset.

	Model							
	mT5				mBART			
Lang	Slot 1	Slot 2	Slot 1&2	TASD	Slot 1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning								
en	$72.15^{\pm 2.04}$	$63.01^{\pm 4.04}$	$54.40^{\pm 3.76}$	$48.62^{\pm 4.08}$	$76.95^{\pm 1.21}$	$76.88^{\pm 1.81}$	$66.87^{\pm 1.93}$	$60.63^{\pm 2.01}$
es	$74.61^{\pm 1.40}$	$71.64^{\pm 0.91}$	$60.83^{\pm 1.80}$	$56.05^{\pm 1.75}$	$75.38^{\pm 1.10}$	$74.91^{\pm 0.61}$	$63.17^{\pm 1.37}$	$58.49^{\pm 1.43}$
fr	$69.15^{\pm 1.60}$	$66.27^{\pm 2.87}$	$51.85^{\pm 2.47}$	$44.56^{\pm 2.51}$	$67.58^{\pm 2.43}$	$70.13^{\pm 3.03}$	$53.28^{\pm 2.54}$	$45.73^{\pm 2.81}$
nl	$65.43^{\pm 3.34}$	$62.44^{\pm 5.28}$	$49.70^{\pm 5.57}$	$43.32^{\pm 5.84}$	$73.10^{\pm 1.44}$	$72.41^{\pm 2.27}$	$59.85^{\pm 2.20}$	$53.12^{\pm 1.90}$
ru	$76.47^{\pm 1.72}$	$69.27^{\pm 2.38}$	$61.61^{\pm 2.53}$	$53.86^{\pm 3.29}$	$76.96^{\pm 1.67}$	$73.07^{\pm 1.00}$	$64.03^{\pm 1.89}$	$56.82^{\pm 1.77}$
tr	$63.87^{\pm 2.73}$	$43.04^{\pm 7.23}$	$30.86^{\pm 5.73}$	$25.37^{\pm 5.65}$	$71.09^{\pm 4.13}$	$58.96^{\pm 4.42}$	$47.75^{\pm 4.61}$	$38.68^{\pm 5.03}$
Prompt-based fine-tuning								
en	$86.46^{\pm 0.90}$	$84.79^{\pm 0.76}$	$76.78^{\pm 0.62}$	$71.67^{\pm 0.57}$	$84.62^{\pm 0.47}$	$85.69^{\pm 0.73}$	$75.82^{\pm 1.02}$	$69.75^{\pm 1.07}$
es	$83.47^{\pm 2.47}$	$78.43^{\pm 2.26}$	$69.31^{\pm 3.27}$	$64.30^{\pm 3.92}$	$81.97^{\pm 1.17}$	$78.75^{\pm 0.86}$	$68.00^{\pm 0.90}$	$61.98^{\pm 0.84}$
fr	$79.76^{\pm 0.58}$	$80.05^{\pm 1.73}$	$66.75^{\pm 1.15}$	$60.05^{\pm 1.34}$	$68.65^{\pm 8.49}$	$69.82^{\pm 11.52}$	$52.68^{\pm 11.39}$	$43.70^{\pm 10.15}$
nl	$81.58^{\pm 0.84}$	$79.19^{\pm 1.58}$	$69.23^{\pm 1.24}$	$62.24^{\pm 1.53}$	$77.03^{\pm 1.87}$	$73.58^{\pm 0.88}$	$62.14^{\pm 0.79}$	$53.63^{\pm 1.17}$
ru	$86.85^{\pm 0.66}$	$82.23^{\pm 0.27}$	$74.94^{\pm 0.81}$	$67.40^{\pm 1.37}$	$84.09^{\pm 0.93}$	$79.88^{\pm 0.88}$	$70.86^{\pm 0.79}$	$62.35^{\pm 0.72}$
tr	$83.14^{\pm 2.28}$	$71.04^{\pm 2.02}$	$59.54^{\pm 3.43}$	$53.27^{\pm 4.27}$	$72.61^{\pm 6.59}$	$61.13^{\pm 5.25}$	$46.94^{\pm 4.26}$	$36.06^{\pm 2.62}$

Table B.2: Monolingual micro F1 scores (in %) of the mT5 and mBART models on different languages with prompt-based and traditional fine-tuning where the examples without any annotation triplets were not filtered from the datasets.

SL	TL	Model							
		mT5				mBART			
		Slot 1	Slot 2	Slot 1&2	TASD	Slot 1	Slot 2	Slot 1&2	TASD
Traditional fine-tuning									
en	es	$60.86^{\pm 4.06}$	$30.65^{\pm 4.57}$	$29.06^{\pm 2.90}$	$25.48^{\pm 2.43}$	$42.68^{\pm 10.34}$	$44.70^{\pm 8.36}$	$29.55^{\pm 5.52}$	$24.87^{\pm 5.08}$
en	fr	$64.68^{\pm 2.57}$	$41.88^{\pm 7.27}$	$33.31^{\pm 5.94}$	$27.90^{\pm 5.74}$	$60.02^{\pm 2.83}$	$58.41^{\pm 4.27}$	$41.54^{\pm 2.64}$	$33.42^{\pm 1.38}$
en	nl	$53.55^{\pm 3.38}$	$31.14^{\pm 2.13}$	$25.36^{\pm 2.58}$	$22.16^{\pm 1.90}$	$63.27^{\pm 1.97}$	$51.28^{\pm 2.33}$	$41.68^{\pm 2.26}$	$36.01^{\pm 1.90}$
en	ru	$54.41^{\pm 8.02}$	$5.41^{\pm 4.38}$	$11.54^{\pm 3.15}$	$9.71^{\pm 2.73}$	$65.65^{\pm 4.00}$	$39.94^{\pm 15.58}$	$34.02^{\pm 9.85}$	$29.68^{\pm 8.65}$
en	tr	$53.66^{\pm 4.98}$	$16.45^{\pm 1.74}$	$10.88^{\pm 1.18}$	$9.88^{\pm 1.34}$	$52.65^{\pm 4.67}$	$31.26^{\pm 7.61}$	$20.90^{\pm 6.93}$	$18.37^{\pm 4.80}$
es	en	$56.16^{\pm 3.33}$	$26.90^{\pm 2.95}$	$26.44^{\pm 2.30}$	$22.84^{\pm 2.11}$	$70.09^{\pm 0.47}$	$61.93^{\pm 3.61}$	$52.89^{\pm 2.68}$	$47.26^{\pm 2.58}$
fr	en	$64.27^{\pm 2.85}$	$39.23^{\pm 4.18}$	$36.13^{\pm 2.92}$	$31.20^{\pm 2.84}$	$71.01^{\pm 2.17}$	$64.40^{\pm 1.95}$	$53.84^{\pm 2.54}$	$46.27^{\pm 4.18}$
nl	en	$57.17^{\pm 2.43}$	$30.45^{\pm 8.91}$	$25.04^{\pm 4.88}$	$21.39^{\pm 4.53}$	$70.78^{\pm 1.50}$	$56.29^{\pm 1.61}$	$48.60^{\pm 2.35}$	$41.84^{\pm 2.43}$
ru	en	$52.92^{\pm 3.44}$	$26.23^{\pm 12.48}$	$27.28^{\pm 10.77}$	$24.65^{\pm 10.90}$	$61.39^{\pm 5.93}$	$52.34^{\pm 10.31}$	$43.54^{\pm 8.63}$	$37.53^{\pm 9.55}$
tr	en	$35.25^{\pm 2.96}$	$13.25^{\pm 5.49}$	$11.07^{\pm 3.40}$	$9.53^{\pm 3.30}$	$55.65^{\pm 10.45}$	$37.65^{\pm 8.97}$	$30.15^{\pm 9.81}$	$24.61^{\pm 9.46}$
Prompt-based fine-tuning									
en	es	$77.89^{\pm 0.83}$	$71.68^{\pm 3.34}$	$60.21^{\pm 2.34}$	$56.29^{\pm 2.27}$	$66.64^{\pm 2.11}$	$58.50^{\pm 6.26}$	$45.69^{\pm 3.97}$	$39.81^{\pm 2.18}$
en	fr	$77.83^{\pm 1.01}$	$73.19^{\pm 0.63}$	$60.73^{\pm 0.83}$	$52.53^{\pm 0.87}$	$66.07^{\pm 2.28}$	$57.23^{\pm 4.46}$	$41.68^{\pm 3.52}$	$32.09^{\pm 2.57}$
en	nl	$77.74^{\pm 1.13}$	$63.57^{\pm 1.09}$	$57.16^{\pm 1.86}$	$52.33^{\pm 1.83}$	$67.80^{\pm 2.69}$	$52.95^{\pm 1.90}$	$44.53^{\pm 1.40}$	$37.48^{\pm 0.75}$
en	ru	$83.54^{\pm 0.50}$	$62.85^{\pm 3.09}$	$58.58^{\pm 2.13}$	$52.68^{\pm 1.66}$	$74.68^{\pm 2.92}$	$58.36^{\pm 8.05}$	$48.13^{\pm 5.88}$	$41.22^{\pm 4.68}$
en	tr	$81.63^{\pm 2.13}$	$52.50^{\pm 3.56}$	$41.35^{\pm 2.72}$	$38.72^{\pm 1.77}$	$57.37^{\pm 5.51}$	$32.83^{\pm 4.16}$	$20.04^{\pm 2.11}$	$16.61^{\pm 2.98}$
es	en	$78.02^{\pm 3.05}$	$63.99^{\pm 7.07}$	$55.30^{\pm 7.02}$	$51.18^{\pm 7.17}$	$74.48^{\pm 2.21}$	$66.39^{\pm 3.28}$	$55.14^{\pm 2.66}$	$47.94^{\pm 1.66}$
fr	en	$84.91^{\pm 0.80}$	$76.11^{\pm 1.77}$	$68.68^{\pm 1.25}$	$64.43^{\pm 1.44}$	$78.94^{\pm 0.97}$	$73.66^{\pm 0.62}$	$61.89^{\pm 0.78}$	$55.11^{\pm 0.79}$
nl	en	$78.80^{\pm 1.57}$	$66.32^{\pm 4.94}$	$58.00^{\pm 4.24}$	$52.80^{\pm 4.53}$	$77.65^{\pm 2.27}$	$66.55^{\pm 2.61}$	$56.98^{\pm 3.41}$	$50.77^{\pm 3.40}$
ru	en	$81.18^{\pm 1.20}$	$73.81^{\pm 3.00}$	$64.84^{\pm 2.19}$	$60.68^{\pm 2.06}$	$76.27^{\pm 1.86}$	$78.63^{\pm 0.73}$	$65.33^{\pm 1.18}$	$59.06^{\pm 0.70}$
tr	en	$76.95^{\pm 2.60}$	$64.19^{\pm 2.32}$	$52.78^{\pm 2.57}$	$47.66^{\pm 1.57}$	$73.02^{\pm 1.56}$	$63.23^{\pm 2.40}$	$50.19^{\pm 1.76}$	$42.29^{\pm 1.53}$

Table B.3: Cross-lingual micro F1 scores (in %) of the mT5 and mBART models with traditional and prompt-based fine-tuning with different combinations of source and target languages (SL and TL, respectively) where the examples without any annotation triplets were not filtered from the datasets.