UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

,,,

MSc thesis in Computer Science

Lukas Muttenthaler

Subjective Question Answering

Deciphering the inner workings of Transformers in the realm of subjectivity

Advisors: Johannes Bjerva, Isabelle Augenstein

Handed in: June 9, 2020

Contents

Abstract

Understanding subjectivity demands reasoning skills beyond the realm of common knowledge. It requires a machine learning model to process sentiment and to perform opinion mining. In this work, I've exploited a recently released dataset for span-selection Question Answering (QA), namely SubjQA [13]. SubjQA is the first QA dataset to date that contains questions that ask for subjective opinions corresponding to review paragraphs from six different domains, namely books, electronics, grocery, movies, restaurants, and TripAdvisor. Hence, to answer these subjective questions, a learner must extract opinions and process sentiment for various domains, and additionally, align the knowledge extracted from a paragraph with the natural language utterances in the corresponding question, which together enhance the difficulty of a QA task. In the scope of this master's thesis, I inspected different variations of BERT [21], a neural architecture based on the recently released Transformer [77], to examine which mathematical modeling approaches and training procedures lead to the best answering performance. However, the primary goal of this thesis was not to solely demonstrate state-of-the-art performance but rather to investigate into the inner workings (i.e., latent representations) of a Transformer-based architecture to contribute to a better understanding of these not yet well understood "black-box" models.

One of the key insights of this work reveals that a Transformer's hidden representations, with respect to the true answer span, are clustered more closely in vector space than those representations corresponding to erroneous predictions. This observation holds across the top three Transformer layers for both objective and subjective questions, and generally increases as a function of layer dimensions. Moreover, the probability to achieve a high cosine similarity among hidden representations in latent space concerning the true answer span tokens is significantly higher for correct compared to incorrect answer span predictions. These statistical results have decisive implications for down-stream applications, where it is crucial to know about why a neural network made mistakes, and in which point in space and time the mistake has happened (e.g., to automatically predict correctness of an answer span prediction without the necessity of labeled data).

Quantitative analyses have shown that Multi-task Learning (MTL) does not significantly improve over Single-task Learning (STL). This might be due to one of the leveraged auxiliary tasks being unsolvable. It appears as if BERT produces domain-invariant features by itself, although further investigation must go into this line of research to determine whether this observation holds across other datasets and domains. Fine-tuning BERT with additional Recurrent Neural Networks (RNNs) on top improves upon BERT with solely one linear output layer for QA. This is most likely due to a more fine-grained encoding of temporal dependencies between tokens through recurrence forward and backward in time, and is in line with recent work.

Chapter 1

Overview

I will begin this thesis with an **Introduction** comprising an overview of the topic being explored. In so doing, I will explain my motivations in conducting this research and outline the importance of continued research on various neural architectures in this field. Following that, I will outline the **Research Questions (RQs)** I aim to answer. Thereafter, in the **Background** section I will introduce the task of **Question Answering (QA)** and discuss to which of the various QA versions I will confine myself in the scope of this master's thesis.

This is followed by an overview of the model architectures that will be leveraged in the different experiments. I will start with explaining the **Transformer**, elaborate on the mechanisms behind **BERT** which is a transformer-based architecture. Moreover, I will discuss the mathematical details with respect to **Recurrent Neural Networks (RNNs)** and **Highway Networks**. To conclude the background section, I will discuss the notion of **Multi-task learning**.

In the **Methodology** section of the thesis, I will explain the different models, the task(s), and most importantly all relevant computations that are necessary to optimize the models concerning the respective task(s).

The elaboration of the methods is followed by a detailed overview of the **datasets** that are exploited to train and evaluate the neural architectures. In this section, I will provide an in-depth analysis of the datasets to both qualitatively and quantitatively assess their nature before any model training.

In the **Quantitative Analysis** section, results concerning all conducted experiments will be presented, explained and discussed. Note that a thorough interpretation of the results will follow in the **Discussion** part and hence interpretation is constrained in this section. Ad hoc elaboration on results may be provided but I refer the interested reader to the **Discussion** section for in-depth interpretations.

Numeric results must be connected to visualizations of models' feature representations in vector space in order to understand the breakthroughs and shortcomings of Machine Learning (ML) models. Hence, an in-depth **Qualitative Analysis** of the hidden representations with respect to selected neural architectures follows the depiction of quantitative results. Alongside this I provide an error analysis to identify the issues the models faced at inference time. Here, I will try to answer **where** along the way and **why** a learner made mistakes.

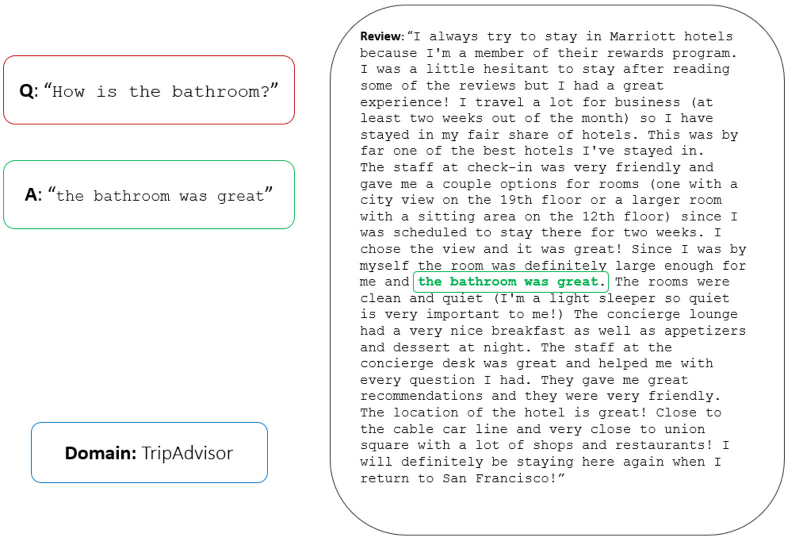
Last but not least, I will **discuss** the results obtained from both types of analyses, draw conclusions and close with a concise **Summary** of the thesis to provide a synopsis free of the hefty details.

Chapter 2

Introduction

Thoroughly understanding the full nature of subjectivity is a daunting task for both humans and machines [8, 59, 83, 84]. Whether it is a subjective thought, an opinion, a question, or an answer, all of it highly depends on the context the respective natural language utterance appears in [52, 84]. It is often not simple to decipher what is and what is not subjective [8, 52]. A question might be subjective but its answer contains an objective, measurable fact, and vice versa [83, 13]. Due to the frequent exchange of opinions in a world greatly embedded in social media, subjectivity in natural language has become highly pervasive. This fact alone makes the task of examining how machines read natural language texts that contain subjective opinions worth pursuing. However, I would like to further stress why I encourage the field of Artificial Intelligence (AI) to shed light on the development of systems that possess the ability to answer questions concerning subjective opinions.

Machine Reading, also called span-selection Question Answering (QA) or Reading Comprehension (RC), has a long-standing history in the fields of Information Retrieval (IR) and Natural Language Processing (NLP). Over the past two decades, of which the last in particular yielded breakthroughs in NLP, machine reading has recorded vast advancements. An array of systems has been developed to enhance machine comprehension systems [75, 70, 82, 81] and numerous different RC datasets have been created to train these [25]. Although much work has been going on in the entirety of open-domain QA [15, 67, 17, 80], I will in this project exclusively focus on the task of finding an answer span in a corresponding natural language context, i.e. span-selection RC.


Q: "How is the bathroom?"

A: "the bathroom was great"

Domain: TripAdvisor

Review: "I always try to stay in Marriott hotels because I'm a member of their rewards program. I was a little hesitant to stay after reading some of the reviews but I had a great experience! I travel a lot for business (at least two weeks out of the month) so I have stayed in my fair share of hotels. This was by far one of the best hotels I've stayed in. The staff at check-in was very friendly and gave me a couple options for rooms (one with a city view on the 19th floor or a larger room with a sitting area on the 12th floor) since I was scheduled to stay there for two weeks. I chose the view and it was great! Since I was by myself the room was definitely large enough for me and **the bathroom was great.** The rooms were clean and quiet (I'm a light sleeper so quiet is very important to me!) The concierge lounge had a very nice breakfast as well as appetizers and dessert at night. The staff at the concierge desk was great and helped me with every question I had. They gave me great recommendations and they were very friendly. The location of the hotel is great! Close to the cable car line and very close to union square with a lot of shops and restaurants! I will definitely be staying here again when I return to San Francisco!"

Figure 2.1: QA example from SubjQA [13]. The correct answer is a text span of n character sequences in the review paragraph corresponding to the subjective question. The span was identified through human crowd-workers before model training. As such, QA is considered a span-selection task, where both start and end position of the correct text span must be predicted by a neural network.

The task of answering questions that contain objective, measurable facts appears to be resolved to a large extent for answerable questions [62, 79]. In so doing, SQuAD v1.0 [62] was the first-ever large-scale dataset that fostered the latter development. As a result, the researchers centered around SQuAD recently developed a more complex dataset that consists of questions that are not answerable, namely SQuAD v2.0 [61]. As can be inferred from the publicly available leader board regarding this task, it appears to be more difficult for models across the board to understand that a question cannot be answered with the given context. This might sound paradoxical at first sight, but if one recalls that humans frequently face the task of acknowledging the fact that in certain cases there simply is no answer, the latter becomes more apparent.

What has been lacking until recently, however, was both a dataset that not only includes unanswerable questions that consist of an objective, measurable fact but is also rich in questions that contain a subjective opinion and a corresponding machine reading system that is capable of understanding and hence answering such questions. At this point, I would like to stress the fact that when I speak about UNDERSTANDING subjectivity, I refer to reading a paragraph and finding the correct answer span within this paragraph (see Figure 2.1 for an example). I am aware of the fact that utterly UNDERSTANDING subjectivity is beyond the scope of current methods in ML [52, 84, 83].

The vast majority of QA datasets is factoid and concerns solely a single domain such as SQuAD v1.0 [62], SQuAD v2.0 [61], WikiQA [85], WikiReading [33] or CNN/Daily Mail [32] of which all but the last dataset are exclusively based on Wikipedia. Recent research in NLP that scrutinized QA datasets revealed that such datasets do not necessarily examine Natural Language Understanding (NLU) abilities, as complex reasoning skills are often not required to perform well [25, 74]. SubjQA, the dataset that I am going to exploit in this study, is the first dataset to date that includes subjective opinions extracted directly from reviews written by humans, does consist of texts corresponding to multiple domains, and includes a high number of unanswerable questions [13]. The latter set of questions has been proven to be particularly difficult since a machine reading system must understand that a question cannot be answered from the given context [61].

Although originality often makes research questions worth pursuing, any research question must come with both a purpose for society and an adequate justification for the pursued avenue. If we, as researchers in the field of ML, develop methods to better understand and analyze how subjectivity and the respective context it appears in is reflected in a neural model, society might benefit from more sophisticated chatbots, search engines, and voice assistants among others shortly. Hence, I will in this thesis contribute to the investigation of both natural language data that contains subjectivity and the behavior of neural machines when faced with the latter as much as time and space allow. In the following sections, I will introduce the general task of QA, both different neural network architectures and training techniques that play a crucial role with respect to the systems that I plan to inspect.

2.1 Related Work

I will, in this work, investigate different neural architectures for span-selection Question Answering (QA) based on the Transformer [77] (see Section 3.1 in the Background section for a detailed elaboration on QA, and Figure 2.1 for a general overview of the task). In so doing, I will inspect different mechanisms (e.g., multi-task learning, sequential knowledge transfer) and training procedures (e.g., adversarial training, different task sampling strategies) to enhance performance concerning subjective QA. In addition, I will look deeper into the inner workings, i.e. hidden representations, of Transformer models at each layer stage. This is done to decipher `how` neural networks answer subjective questions, and unveil `where` along the way and `why` they make mistakes. A thorough qualitative analysis of model behavior appears crucial, given that deep neural networks (DNNs) are often considered "black-box" models that require better understanding by the community [43].

The recent advent of Transformer models [77] and NLP models based on the latter such as ELMo [57], BERT [21], and RoBERTa [48], has yielded an enormous flux of studies that draws attention to NLP in general and open-domain QA in particular. One recent study that is similar to this work conducted a layer-wise analysis of BERT's Transformer layers to investigate how BERT answers questions [2]. For each of BERT's

Transformer layers, they projected the model's high-dimensional hidden representations into $\mathbf{R}^2$ to visually depict how BERT clusters different parts of an input sequence (i.e., question, context, answer) while searching for the correct answer span in latent space. The main difference, however, is that the aforementioned study exclusively conducted a qualitative analysis of BERT's hidden representations without the endeavor to implement different model versions to quantitatively inspect performance concerning QA. Moreover, BERT was fine-tuned on factoid and not on subjective questions which most likely yields different QA behavior and hence different feature representation patterns in latent space as both opinions and sentiment are more relevant than objective, measurable facts to answer a subjective question. Their attempt to explain QA behavior through thoroughly analyzing BERT's hidden representations at various layer stages was, nevertheless, remarkable and a crucial step forward towards explainability in AI which is why I will follow their approach concerning the qualitative analysis of feature representations in vector space, and inspect whether their results are replicable for the realm of subjectivity.

Another study that has been published recently, developed a dataset that significantly differs from most recent QA datasets. As mentioned at the beginning of this section, the vast majority of QA datasets is factoid and concerns only a single domain [62, 85, 32]. Their dataset followed the attempt to explicitly avoid questions that may be answered with common knowledge or knowledge about one domain [25]. This attempt follows a similar motivation for the development of SubjQA [13]. They created a dataset that consists of four domains of which two cannot be answered with pre-training on corpora that contain common knowledge (e.g., "During which period was Bill Clinton president of the United States of America?"). However, the dataset is with 800 texts not particularly large and does not include any questions with respect to subjective opinions of humans. Moreover, the study exclusively focused on dataset development and analysis without looking into the behaviour of SOTA NLP models while answering questions. The latter is decisive to both understand how neural networks process natural language utterances contained in the dataset which potentially yields insights into the quality of the respective dataset, and whether human annotations are reliable sources.

Arkhangelskaia et al., 2019 [5] investigated which tokens in question - context sequence pairs receive particular attention by BERT's self-attention mechanisms to answer a question, and how the multi-headed attention weights change across the different layers in BERT. Similarly to [2], the authors did solely conduct a qualitative analysis of the model. Contrary to [2], the study focused on a single implementation of BERT and exclusively exploited SQuAD [62, 61] without inspecting BERT's behaviour with respect to other, more challenging QA datasets where contexts belong to different domains.

Both Bingel & Søgaard, 2017 [9], and Bjerva, 2017 [12] investigated the relatedness between auxiliary and main tasks in different multi-task learning (MTL) paradigms. They analyzed the importance of relations between tasks, and under which conditions auxiliary tasks unfold to be beneficial for the main task(s). However, they exclusively scrutinized classification and structured prediction tasks. Structured prediction tasks are tasks, where the model is meant to predict symbols rather than real values. In NLP such tasks are summarized under the umbrella term of sequence tagging or labeling (e.g., Part-of-Speech tagging, Named Entity Recognition). Moreover, they solely deployed Recurrent Neural Networks (RNNs) [26]. In contrast, I will for the first time investigate different MTL settings for span-selection QA and in so doing leverage the knowledge of a pre-trained Transformer model [77], namely BERT [21].

Numerous studies have worked on the development of QA datasets or put effort into the advancement of RC models that perform well on them. Little, however, has gone into the examination of non-factoid questions, which are questions that cannot be answered with objective, measurable facts but require models to understand subjectivity without resorting to common knowledge that might be present in large pre-training corpora. BERT is complex enough to perform incredibly well on factoid questions that correspond to paragraphs from a single domain and frequently require common knowledge to be answered [21, 79]. Whether BERT does also achieve close to human performance on datasets that barely consist of factoid questions, require little common knowledge, and contain texts from multiple domains is yet to be deciphered and will play a major role in the current study.

2.2 Research Questions

In this study, I will examine the following research questions (RQs) as thoroughly as space and time allow.

1. `Is` it necessary to fine-tune a Transformer model on a span-selection QA dataset that consists of subjective questions and multiple domains, namely SubjQA [13], to achieve SOTA performance with respect to the latter? Or is it sufficient to leverage the knowledge of a pre-trained Transformer model, namely BERT [21], that was previously fine-tuned on SQuAD? SQuAD is a span-selection QA dataset that exclusively contains objective questions with respect to a single domain [62, 61].

2. `Do` additional encoding layers that are not exploited by neural architectures based on the Transformer [77], such as Long-Short Term Memories (LSTMs) [35] or Highway Networks [73], on top of BERT enhance information processing to an extent such that QA performance is increased? This research question is based on one recent study that has shown that encoding temporal dependencies among tokens through Recurrent Neural Networks (RNNs) [26] helps BERT for QA [37]. However, this was the first study to date that has investigated the latter and their experiments were performed solely with respect to SQuAD. Thus, it is both worth replicating their results and analysing whether this hold for subjective questions too.

3. `To` which extent does multi-task learning (MTL) and adversarial learning techniques enhance BERT's answering behaviour with respect to subjective questions corresponding to multiple domains? In so doing, I will leverage the crowd-sourced human labels concerning the degree of subjectivity [13], and adversarial training methods such as Gradient Reversal Layers (GRLs) [27].

4. `How` does model performance differ as a function of review domains? Since SubjQA contains review paragraphs corresponding to multiple domains it appears crucial to investigate a learner's performance with respect to these domains, and inspect whether some domains are more difficult than others.

5. `Is` it possible to infer the difficulty of subjective questions from the interrogative words (e.g., `how`, `what`, `which`) that introduce them? If so, `how` does the degree of difficulty differ among them?

6. `Which` qualitative insights can be extracted from a Transformer's hidden representations in latent space? Is it possible to decipher `where` along the way (i.e., in which layer) a learner made mistakes to better understand `why` an answer span was not predicted correctly? Moreover, is there a difference between subjective and objective questions with respect to a model's answer span search in latent space? If there is a difference, `which` is it, and `how` could this insight be leveraged?

Chapter 3

Background

3.1 Question Answering

The task of Question Answering (QA) has a long-established history in the connected fields of Information Retrieval (IR) and Natural Language Processing (NLP). Since this thesis is centered around a research project in NLP, I will confine myself to the position of QA within the latter research area. Hence, the role of search as a crucial part in finding relevant documents, which mainly belongs to the realm of IR, will not be discussed here. The focus of the thesis lies in reading and not retrieving. There are different versions of QA, namely closed-domain and open-domain QA. Since I will exclusively focus on open-domain QA, I will briefly introduce the former and elaborate more thoroughly on the latter.

Closed-domain QA deals with questions that concern a specific domain [76, 22, 10], e.g., chemistry, pharmacy or neurology. This can be useful if one aims at analyzing numerous research papers in one of the aforementioned fields to either write a review paper or conduct a meta-analysis, or would like to perform plagiarism detection concerning a certain domain. Closed-domain QA, however, is restricted to a confined domain, does not require common knowledge, and is forced to exploit notably smaller datasets than open-domain QA, as resources are sparse. Hence, closed-domain heavily relies on ontologies such as knowledge graphs which often contain a large amount of factual information.

Open-domain QA was initially defined as finding answer spans in collections of unstructured, raw text [17]. In open-domain QA, a Machine Comprehension (MC) system must retrieve the relevant documents to answer the respective question [15, 67, 17, 80]. This is mainly performed through IR search methods. In so doing, the system is first required to understand which documents are decisive to perform the latter step. After finding those documents, the system has to search for the correct text span within the corresponding natural language paragraphs to correctly respond to the question. In earlier versions of open-domain QA, structured data such as ontologies, databases or Knowledge Bases (KBs) such as the popular Freebase KB [14] were frequently exploited to develop and evaluate QA systems (e.g., [86]). Due to their limitations and expensiveness, however, recent research in QA has shifted its attention again towards finding answers in pieces of raw text rather than retrieving information from KBs. Interestingly, this was the focus of QA in the first place. In this study, I will exclusively draw attention to documents of unstructured, raw natural language text without the utilization of KBs.

Span-selection QA, or RC, may be perceived as a sub-field of open-domain QA, and will be the focus of this thesis. As such RC is concerned with the development of Machine Reading (MR) systems that are capable of finding an answer span in the documents that were previously retrieved via IR methods. Recall that the latter represents the first step in any open-domain QA setting. The systems are meant to first read a short paragraph regarding a certain domain about which a question is asked. As a next step, they must find the correct answer span in the paragraph whenever a question is answerable. There are cases where a question cannot be answered given the corresponding context [61]. The correct answer then simply evaluates to the empty string. The latter problem has been addressed only recently and remains open to investigation. To foster an examination in this area, similarly to SQuAD v2.0 [61], a significant number of questions in SubjQA is unanswerable [13].

3.2 Transformers

The recent advent of the Transformer [77] has yielded an enormous deluge of studies concerning neural architectures, particularly in NLP. Never before has NLP received as much attention as since the release of the Transformer. Among the best-performing Natural Language Understanding (NLU) systems are solely models that leverage this neural architecture in one way or another, as can be inferred from publicly available leaderboards such as GLUE [79] [1], SuperGLUE [78] [2], SQuAD v1.0 [62] [3], SQuAD v2.0 [61] [4], or WikiSQL [88] [5]. Owing to this recent development in NLP I will exclusively exploit architectures that are based on the Transformer. Hence, I will now provide a short introduction about the general concept and the mathematical details behind this neural model.

Contrary to Recurrent Neural Networks (RNNs) [26] and Convolutional Neural Networks (CNNs) [44], the Transformer [77] does neither leverage temporal dependencies between timesteps through recurrence nor spatial features through filters and sliding windows respectively. The architecture is exclusively based on attention mechanisms, first introduced by Bahdanau et al., 2014 [7], which makes it highly parallelizable and computationally efficient. As such, it does not suffer from the same memory constraints as sequential computation does. The main advantage of exploiting attention mechanisms is that dependencies between timesteps (e.g., tokens) can be modelled independent of their position in the input sequence or distance to a token in the output sequence such as in Machine Translation (MT) where learning dependencies between distant positions is indispensable to correctly map words from one language to another [7, 50, 77].

In sequential computation an input sequence is summarized as the recursively computed hidden representation of which each element at timestep t is dependent on both the hidden representation at the previous timestep $t-1$ and the current input at t (this is explained in more detail in Section 3.3). In so doing, all inputs are weighted equally. This bears the constraint that early positions in an input sequence may be overwritten by later positions or simply weighted less. In the worst case, this can lead to a phenomenon called catastrophic forgetting and hence result in an enormous performance decline for longer sequences [7, 50]. In contrast, self-attention or intra-attention mechanisms possess the ability to relate positions in an input sequence independent of their distance to each other, yielding a more informative and richer representation of an input sequence [7, 50, 77, 21]. This alleviates the aforementioned constraint that models solely based on recurrence suffer from. The Transformer is the first neural architecture to date that exclusively leverages such self-attention mechanisms without utilizing any recurrence or convolutions in its computation of sequence representations. As such, it only consists of self-attention and point-wise, fully connected layers stacked on top of each other.

The Transformer's self-attention mechanisms employ a scaled version of the dot-product attention, first introduced in [50]. The difference between dot-product and scaled dot-product attention is the scaling factor $\frac{1}{\sqrt{d_k}}$, where d_k corresponds to the dimensions of query and keys, and was introduced to counteract the potential vanishing gradient problem the softmax function might suffer from when d_k becomes large [77].

The input to the attention function contains queries and keys of dimension d_k and values of dimension d_v. The sets of query, keys, and values, are represented as the three matrices Q, K, and V, and simply evaluate to linear layers (see Figure 3.1). To obtain attention weights, a softmax function - the softmax function will be explained in more detail in Section 4 but basically maps a vector of continuous values to a discrete probability distribution - is applied to the inner product of Q and K, scaled by $\frac{1}{\sqrt{d_k}}$.

The matrix of attention weight vectors is then multiplied with V to obtain a weighted version of the values V (see leftmost rectangle in Figure 3.1). The computation can be summarized as follows,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3.1}$$

[1] https://gluebenchmark.com/leaderboard/

[2] https://super.gluebenchmark.com/

[3] https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/

[4] https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/

[5] https://github.com/salesforce/WikiSQL

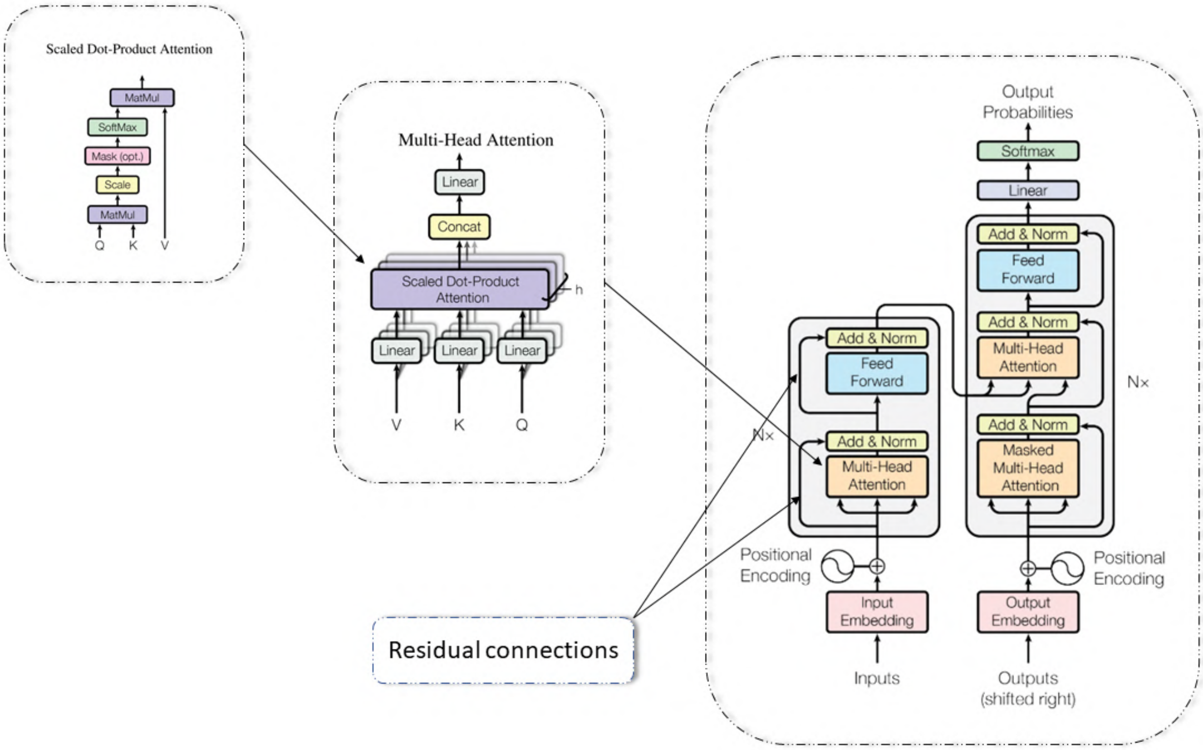


Figure 3.1: Scaled Dot-Product Attention, Multi-Head Attention and the Transformer architecture [77].

This attention function is performed h times in parallel (see center rectangle in Figure 3.1) instead of just once. The h output values, each of dimension d_v, are then concatenated, and passed through another linear layer. The outputs of this self-attention layer serve as inputs to a point-wise, feed-forward neural network, together forming a single Transformer block (see rightmost rectangle in Figure 3.1).

A Residual connection [31] is applied to each sub-layer, followed by layer normalization [6]. Residual connections contribute to richer feature representations through adding function inputs to function outputs [31]. Layer normalization significantly reduces the training time of deep neural architectures through stabilizing activities of individual neurons within layers, thus making a model computationally more efficient [6].

Based on this architecture, a large number of neural Language Models (LMs) has been developed recently to tackle an array of problems in NLP, and in so doing outperforming more traditional approaches. Of those, BERT [21] has been proven to be the most successful, and thus received ample attention lately. I will now explain the mechanisms behind BERT, and elaborate on why it is more successful than other architectures.

3.2.1 BERT

BERT refers to Bidirectional Encoder Representations from Transformers and as such is the first deeply bidirectional Language Model (LM) based on the Transformer architecture. Traditionally, the objective of pre-training LMs was to predict words given **either** the right or the left context of some window size (e.g., previous n-gram) [63]. This is called left-to-right or right-to-left language modelling. Until the advent of BERT, all LMs based on the Transformer or Long Short-Term Memories (LSTMs) [35] were deployed either unidirectional [60] or shallowly bidirectional [56, 57], and therefore not capable of contextualizing a word given the entire context the word appears in (i.e., right- and left-hand side of a token). BERT, however, has closed the gap and, as the name suggests, exploits the context both to the left and right of a word (see Figure 3.2 for a comparison between BERT and the aforementioned models with respect to their pre-training). BERT is thus the first unsupervised, deeply bidirectional LM for NLP that exclusively leverages fully connected linear layers and self-attention mechanisms that can easily relate tokens independent of their positions in an input sequence [21]. This is particularly important for token-level tasks such as QA, where the context to both the left and right of an input token is decisive to find the correct answer span in a paragraph. Hence, BERT became indispensable in the disentanglement of a word's context on a variety of NLP tasks, as numerous recent studies have shown [18, 28, 87], and both the GLUE and SuperGLUE leaderboards indicate [79, 78], where models that deploy BERT, or optimized versions of BERT (e.g., [48, 42]), clearly outperform more traditional approaches.

BERT's main pre-training objective is masked language modelling (MLM) [21]. That is, some of the tokens of an input sequence are randomly masked, and the model is optimized to infer their vocabulary IDs based solely on their contexts. In contrast to standard left-to-right LM pre-training (e.g., [60]), BERT is optimized to jointly condition on both directions. As a result, fine-tuning for downstream application can easily be deployed, and requires nothing more than one additional task-specific output layer [21] (see Figure 3.3). The inputs for such linear output layers are BERT's deeply bidirectional feature representations corresponding to an input token sequence, yielded through the pre-trained MLM objective. There is, however, the possibility to inform BERT about temporal dependencies through leveraging recurrence during fine-tuning, which has recently been explored with respect to token-level tasks [37]. I will investigate further into this idea and scrutinize whether additional recurrent layers on-top of the pre-trained BERT model enhance performance concerning QA. This might fuse the best of both worlds for sequence modelling tasks: using a highly parallelizable and computationally efficient Transformer during pre-training, and exploiting recurrence during fine-tuning via (bidirectional) LSTMs.

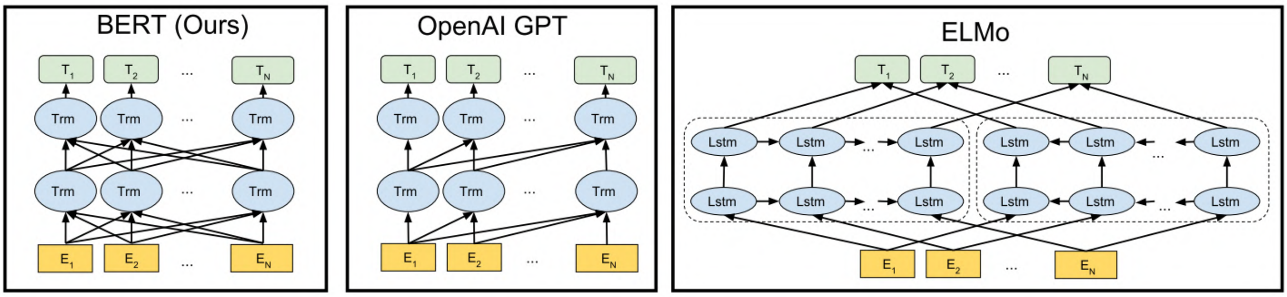
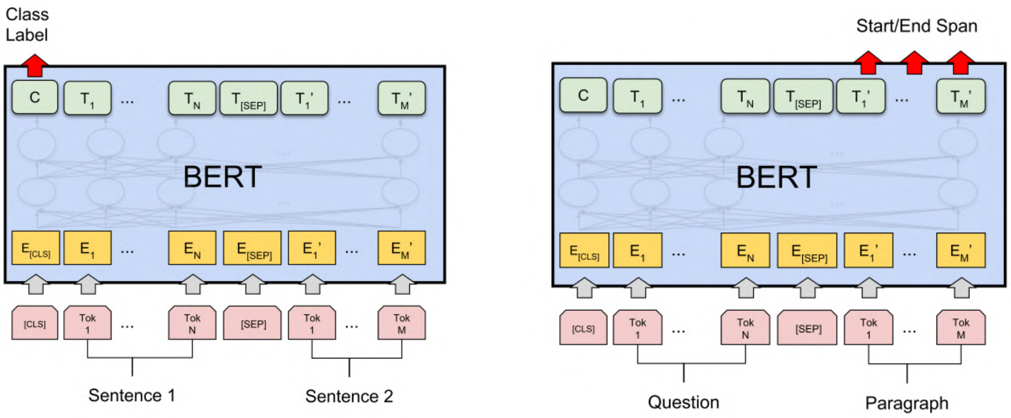


Figure 3.2: Different pre-training techniques. BERT [21] leverages a deep bidirectional Transformer. Open AI's GPT [60] exploits left-to-right Transformers. ELMO [56, 57] uses the concatenation of forward (left-to-right) and backward (right-to-left) LSTMs. As one can infer from the figures, solely BERT is jointly conditioned on the left and right context across layers. Figure copied from [21].



(a) BERT fine-tuning for sentence pair classification.

(b) BERT fine-tuning for QA.

Figure 3.3: BERT fine-tuning [21].

3.3 Recurrent Neural Networks

I will now briefly explain the algorithm behind bidirectional RNNs as this is another neural architecture I exploit in the experiments. As their name suggests, bidirectional RNNs recursively process a word sequence $\mathbf{x}_i^T = [x_i^1, x_i^2, ..., x_i^T]$ forward and backward in time [69]. As such, both past-aware $\overrightarrow{\mathbf{h}_i} = [h_i^1, h_i^2, ..., h_i^T]$ and future-aware $\overleftarrow{\mathbf{h}_i} = [h_i^T, h_i^{T\text{-}1}, ..., h_i^1]$ hidden representations of an input sequence $\mathbf{x}_i^T$ are computed in every recurrent hidden layer h_i.

$$\overrightarrow{\mathbf{h}}_i^{(t)} = \mathrm{LSTM}\left(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{z}_t\right), t = 1, \cdots, |x| \tag{3.2}$$

$$\overleftarrow{\mathbf{h}}_i^{(t)} = \mathrm{LSTM}\left(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{z}_t\right), t = |x|, \cdots, 1 \tag{3.3}$$

To compute the final hidden state $\mathbf{h}_i$, forward $\overrightarrow{\mathbf{h}_i}$ and backward $\overleftarrow{\mathbf{h}_i}$ hidden states are summed, before passing the sequence further to the next layer. The latter is done to keep the dimensionality of feature representations constant. Recall that $\mathbf{z} \in \mathbf{R}^{768}$. [6] If we pass $\mathbf{z}$ through a BiLSTM and compute hidden representations both forward and backward in time of which both $\{\overrightarrow{\mathbf{h}}_i^{(t)}, \overleftarrow{\mathbf{h}}_i^{(t)}\} \in \mathbf{R}^{768}$, a concatenation would thus yield $\mathbf{H}_i^{(t)} \in \mathbf{R}^{768\times 2}$ which we would like to avoid to both not overload the computational budget and keep the dimensionality of feature vectors in the same $\mathbf{R}$ space. Otherwise the comparison might not be equal as we would increase the dimensionality of feature representations twofold which could potentially result in more fine-grained word embeddings. Hence, $\mathbf{H}_i^{(t)}$ is computed as follows.

$$\mathbf{H}_i^{(t)} = \overrightarrow{\mathbf{h}}_i^{(t)} + \overleftarrow{\mathbf{h}}_i^{(t)} \tag{3.4}$$

Since LSTMs in contrast to vanilla RNNs or GRUs [20] consist of both hidden and cell states, the same computation as outlined above must be performed for cell states $\mathbf{c}_i$. LSTM denotes the LSTM function [35], $\mathbf{h}_i^{(t)}$ is the hidden state at time step t, and $\mathbf{z}$ represents the contextual word embedding yielded by BERT, that is $\theta(\mathbf{x}) \in \mathbf{R}^{768}$.

Before moving to the second POST-BERT encoding neural architecture, I would like to discuss a potential caveat of such recurrent encodings. Recall that the special [CLS] token in BERT reflects the semantic representation of an entire word sequence $\mathbf{x}_i$ (see Figure 3.3). Speaking with respect to temporal dependencies, the latter is the token at timestep 0, or in other words the token at the 0^{th} index of the word vector in latent space. An RNN, however, recursively processes a sequence of words timestep by timestep, starting at 0 and stopping at T, taking into account both each previous hidden state h_i^{t-1} and each current input z_i^t. Hence, the semantic representation of a sentence is not encoded at the 0^{th} index as in BERT but at the last index, that is at timestep T. This does not come with any problems for a QA task, where one has to pass the entire hidden representation of an input sequence to the fully-connected QA output layer. It might even be beneficial, as one recent study has shown, due to the fact that temporal dependencies are additionally encoded after BERT [37]. For simple classification tasks, however, this might indeed result in a problem. Usually, one is required to exclusively pass the hidden feature representation encoded in the [CLS] token to a classification output layer (see Figure 3.3). If one applies a recurrent neural module before employing the latter step, then the question of which latent representation must be used for the classification layer becomes non-trivial. By virtue of simplicity and according to common knowledge about BiLSTMs [35, 69], I will leverage the hidden representation at the last time step as the input vector for classification layers - when performing STL classification experiments or MTL.

3.4 Highway Networks

Network depth comes with the cost of longer training cycles, the necessity of more training data, and the complexity of more sophisticated optimization and regulation techniques to exchange information between different

[6] For simplicity, for now $\theta(\mathbf{x}) = \mathbf{z}$.

layers [29, 43]. Although powerful and highly promising with respect to machine learning tasks, such deep networks require careful training procedures. Highway networks were developed to facilitate this information flow through employing some form of regulation between layers in deep neural architectures [73]. In so doing, Highway networks employ gating units to regulate the flow through the network. This is deployed in a similar manner as the gating mechanisms in LSTMs [35]. Those gates enable paths along which information flows across layers, namely information highways. Hence, the name Highway networks.

In general, a feedforward linear layer is deployed as follows,

$$\mathbf{y} = H\left(\mathbf{z}, \mathbf{W_H} + \mathbf{b_H}\right). \tag{3.5}$$

This layer is called the projection layer and is denoted with the letter H. It employs an affine transformation on BERT's latent feature representation $\mathbf{z}$, followed by a rectified linear unit (ReLU) non-linearity which sets all negative values of an input vector or matrix to 0[7] to yield the projection layer's output $\mathbf{y}$. The weights of this layer $\mathbf{W_H}$ are initialized according to the Xavier uniform initialization, also called Glorot initialization [29], which generally yields better results than a simple random initialization of a layer's weights. In a Highway network, two linear transforms are employed in addition to H, a transform gate $T\left(\mathbf{z}, \mathbf{W_H}\right)$ and a carry gate $C\left(\mathbf{z}, \mathbf{W_H}\right)$, thus resulting in the following equation,

$$\mathbf{y} = H\left(\mathbf{z}, \mathbf{W_H} + \mathbf{b_H}\right) \odot T\left(\mathbf{z}, \mathbf{W_T} + \mathbf{b_T}\right) + \mathbf{z} \odot C\left(\mathbf{z}, \mathbf{W_C} + \mathbf{b_C}\right). \tag{3.6}$$

In contrast to the projection layer H, which is followed by a ReLU non-linearity, the transform gates T and C are both followed by a sigmoid function, that is

$$\sigma(\mathbf{z}) = \frac{1}{1 + e^{-\mathbf{z}}}, \mathbf{z} \in \mathbf{R}, \tag{3.7}$$

where

$$\mathbf{z} \in \{(\mathbf{z}, \mathbf{W_T} + \mathbf{b_T})\,; (\mathbf{z}, \mathbf{W_C} + \mathbf{b_C})\}, \tag{3.8}$$

instead of

$$\mathbf{ReLU}\left(\mathbf{z}, \mathbf{W_H} + \mathbf{b_H}\right) = \max(0, (\mathbf{z}, \mathbf{W_H} + \mathbf{b_H})). \tag{3.9}$$

According to the original paper [73], C is set to $C = 1 - T$ and therefore also employed in the current set-up. Hence,

$$\mathbf{y} = H\left(\mathbf{z}, \mathbf{W_H} + \mathbf{b_H}\right) \odot T\left(\mathbf{z}, \mathbf{W_T} + \mathbf{b_T}\right) + \mathbf{z} \odot \left(1 - T\left(\mathbf{z}, \mathbf{W_T} + \mathbf{b_T}\right)\right). \tag{3.10}$$

One can quickly see that this is nothing other a sum between an element-wise multiplication (i.e., Hadamard product) between two feedforward neural networks and an element-wise multiplication (i.e, Hadamard product) between an input sequence $\mathbf{z}$ and another, differently parametrized feedforward neural network. The fact that this transformation not only takes into account the linear transformations yielded by the different gates but also the input $\mathbf{z}$, similar to a residual block, makes a Highway network more flexible and thus better suited for networks with more depth than a simple linear transform [73]. Furthermore, a Highway network has thus the ability to adaptively transform or copy feature representations which might be beneficial as a bridging step between BERT and an output layer for the final classification. The dimensionality for both inputs, outputs, layers, and gates must be the same in a Highway block consisting of fully connected layers, which is what I employ. Hence, $\{\mathbf{z}, \mathbf{y}, H(\mathbf{z}, \mathbf{W_H} + \mathbf{b_H}), T(\mathbf{z}, \mathbf{Wf_T} + \mathbf{b_T})\} \in \mathbf{R}^{768}$.

[7] $f(z) = \max(0, z)$

3.5 Multi-task Learning

In multi-task learning (MTL) a learner is required to perform several tasks in parallel. Hence the name MTL. This is in stark contrast to single task learning (STL), where a model is optimized to perform well on a single task only. In MTL, one of the tasks usually serves as the main task, which a learner is evaluated on at inference time, and the remaining tasks serve as auxiliary tasks to provide useful information that enrich a model's feature representations to enhance performance on the main task [16, 11]. In general, the learner, e.g., a neural network, is trained on a training set D_t, that is exploited across all tasks in the task set $\mathbf{T}$. Hence, a main task T and an auxiliary task T' are drawn from the same training set D_t but leverage different signals to perform well on the respective tasks.

As such, MTL may be considered an inductive transfer method that exploits the domain specific information in training signals across different tasks and therefore introduces noise that help a model to generalize better with respect to unseen data [16]. MTL has recently received ample attention in Deep Learning (DL) in general and NLP in particular owing to its success across a variety of machine learning tasks [65, 47]. However, the benefits of MTL unfold if and only if the tasks in the task set $\mathbf{T}$ are related [12, 4, 9]. That means, that a task T and another task T' must have a somewhat similar training objective, where the signals from T' are beneficial to enhance a model's performance concerning T. If the latter is not guaranteed, a model will learn feature representations that are not useful for any of the tasks in $\mathbf{T}$ [16, 12, 9]. What makes the latter set-up particularly computationally efficient is the fact that MTL is exclusively deployed during training. At inference time, the model is solely evaluated on the main task T as this is the model's primary objective.

In MTL for neural networks, all feature representations F in a model's set of hidden layers $\mathbf{H}$ are shared across tasks, whereas the model contains task-specific parameters for each task in $\mathbf{T}$ that are not shared between tasks and are part of the output layer corresponding to each task. This is called HARD PARAMETER sharing. HARD PARAMETER sharing has proven to be more successful than SOFT PARAMETER sharing for a variety of machine learning tasks, and is therefore considered common practice in MTL in general [16] and for NLP in particular [11, 65, 47]. Moreover, HARD PARAMETER sharing is computationally more efficient than SOFT PARAMETER sharing, where each task in $\mathbf{T}$ has its own model and corresponding hidden feature representations F. The number of models is equal to the number of tasks, and thus the number of parameters in the shared parameter set θ is as many times higher than in HARD PARAMETER sharing as there are tasks in $\mathbf{T}$ and as such requires notably more computational budget. What is more, HARD PARAMETER sharing is easier to implement and vastly reduces the risk of overfitting on the main task T as a model has to share its feature representations across tasks [9]. This might not hold for SOFT PARAMETER sharing owing to the fact that the feature representations F are shared across models but do not necessarily force a single model to produce feature representations that are useful for all tasks in $\mathbf{T}$. SOFT PARAMETER sharing is employed during sequential knowledge transfer, where a model is sequentially optimized until convergence with respect to each task in $\mathbf{T}$ [16]. This is opposed to parallel transfer, where a learner is simultaneously trained on all tasks. I will, due to the aforementioned reasons, primarily draw attention to HARD PARAMETER in the experiments but also employ SOFT PARAMETER sharing to contrast MTL techniques against each other.

Chapter 4

Methodology

4.1 Model

Figure 4.1 illustrates the deployed MTL architecture, and serves as an introductory high-level overview. I will now go through each part of the model step-by-step from bottom to top, starting with extracting contextual features through BERT [21], a neural architecture based on the Transformer (see Section 3.2).

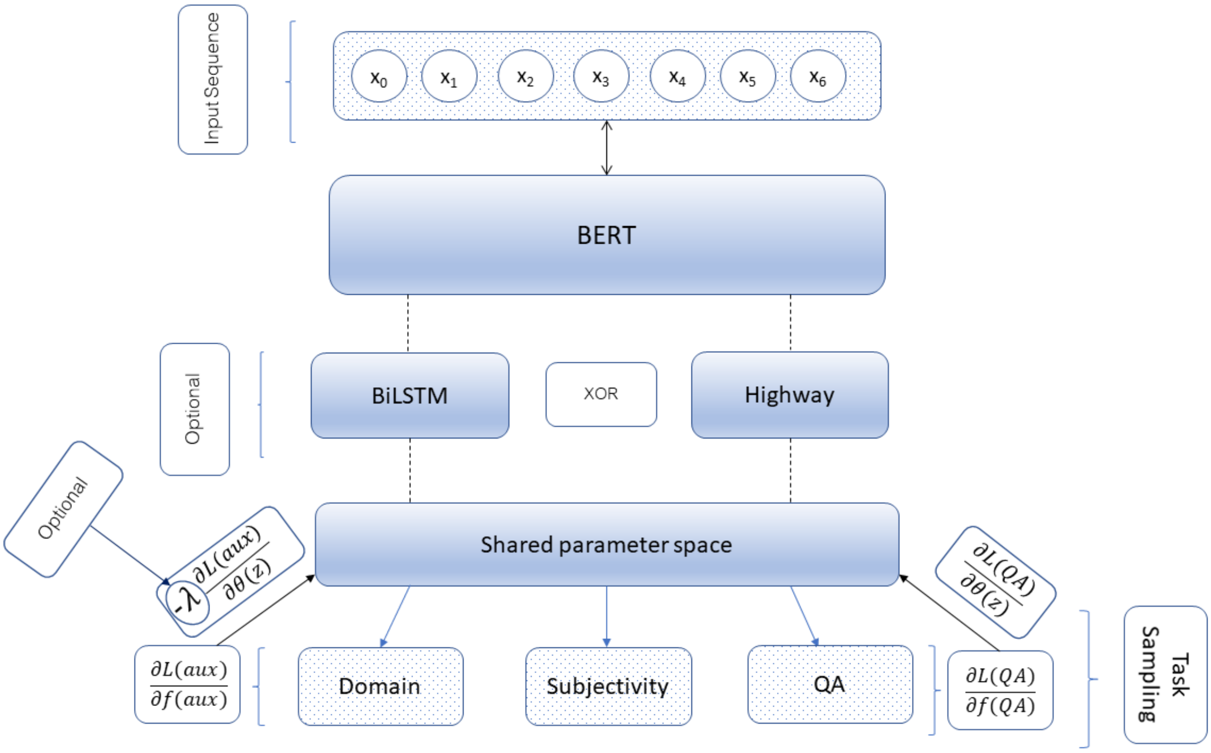

Input Sequence
x_0
x_1
x_2
x_3
x_4
x_5
x_6
BERT
Optional
BiLSTM
XOR
Highway
Optional
Shared parameter space
$-\lambda$
$\frac{\partial L(aux)}{\partial \theta(z)}$
$\frac{\partial L(QA)}{\partial \theta(z)}$
$\frac{\partial L(aux)}{\partial f(aux)}$
Domain
Subjectivity
QA
$\frac{\partial L(QA)}{\partial f(QA)}$
Task Sampling

Figure 4.1: Multi-task learning for QA. Additional POST-BERT encoding layers are optional. Feature representations are shared across all tasks (i.e., HARD PARAMETER sharing). Each task in the task set $\mathbf{T}$ has its own task-specific output layer. The backprogated gradients w.r.t. the auxiliary tasks may be reversed. The latter depends on the employed MTL set-up. For simplicity, the input sequence x_i^T is depicted in $\mathbf{R}^7$.

4.1.1 BERT

For every implemented QA model, a pre-trained DISTILBERT Transformer [68] serves as the feature extractor prior to any task-specific output or POST-BERT custom encoding layers [1]. Compared to BERT [21], of which the Base and Large model consist of 12 and 24 Transformer layers respectively, DISTILBERT only contains 6 Transformer layers without showing a statistically significant deterioration in performance compared to BERT Base on a variety of NLP downstream tasks [79, 68]. This makes DISTILBERT highly user-friendly and easy to deploy.

Updating the weights of a full BERT Transformer model is not feasible with the available computational budget. Hence, I am required to leverage a distilled version of BERT. For simplicity and to facilitate reading, I use the name BERT throughout the following sections when referring to a DISTILBERT model.

4.1.2 Notations

Table 4.1 depicts the notations that will be used throughout the following sections.

Math notation	Natural Language reference
$(\mathbf{q}, \mathbf{c})$	question - context pair
$(\mathbf{q}, \mathbf{a})$	question - answer pair
$\mathbf{x}$	input sequence
$\tilde{\mathbf{f}}$	a model
$\theta(\mathbf{x})$	feature extraction through BERT
ϕ	additional encoding/fine-tuning layer(s)
f_k	task-specific output layer
f_{qa}	QA head
f_{sbj}	subjectivity head
f_{dom}	context-domain head

Table 4.1: Notations that will be used throughout the following section(s).

4.1.3 Multi-task Learning

On top of the pre-trained BERT language model (LM) which I refer to as $\theta(\mathbf{x})$ throughout the following paragraph, I build my own task-specific output layers. Each model $\tilde{\mathbf{f}}$ consists of a fully-connected feed-forward linear output layer which I refer to as $f_{qa}(\theta(\mathbf{x}))$ to predict the answer span $\mathbf{a}$ within the context $\mathbf{c}$ given an input question-context pair $(\mathbf{q}, \mathbf{c}) = \mathbf{x}$. In addition to QA, which $\forall\tilde{\mathbf{f}}$ is implemented as the main task, neural networks are augmented with $\mathbf{k}$ auxiliary task modules, where either $\mathbf{k} = 1$ or $\mathbf{k} = 2$. I refer to the task-specific output layer corresponding to the classification of both a question and its respective context $(\mathbf{q}, \mathbf{c})$ into a subjective opinion vs. an objective, measurable fact as the classifier $f_{sbj}(\theta(\mathbf{x}))$. Hence, the first auxiliary task is defined as a binary sequence classification task with respect to both the question $\mathbf{q}$ and its corresponding answer $\mathbf{a}$.

On the other hand, the second auxiliary task $\forall\tilde{\mathbf{f}}$ with $\mathbf{k} = 2$ auxiliary modules is defined as a multi-way classification of the CONTEXT-DOMAIN $\mathbf{c}^d$, where the number of classes is 6 or 7 respectively, dependent on whether $(\mathbf{x}, \mathbf{y}^d) \in D_{Subj}$ or $(\mathbf{x}, \mathbf{y}^d) \in D_{Subj} \cup D_{SQuAD}$[2]. A domain, $\mathbf{y}^d$, is part of the following set, $\{\text{BOOKS}, \text{ELECTRONICS}, \text{GROCERY}, \text{MOVIES}, \text{RESTAURANTS}, \text{TRIPADVISOR}, \text{WIKIPEDIA}\}$. The latter domain, namely WIKIPEDIA, is part of the class set, if and only if $(\mathbf{x}, \mathbf{y}^d) \in D_{Subj} \cup D_{SQuAD}$. I refer to task-specific output layers for the latter auxiliary task as $f_{dom}(\theta(\mathbf{x}))$.

There is, however, the possibility that the model hierarchy does not evaluate to the aforementioned $f_k(\theta(\mathbf{x}))$ structure but must rather be depicted as $f_k(\phi(\theta(\mathbf{x})))$, where ϕ summarizes the parameter set of custom NN

[1] https://huggingface.co/transformers/

[2] D_{Subj} and D_{SQuAD} refer to question-review pairs from SUBJQA or SQUAD respectively.

encoders on top of $\theta(\mathbf{x})$. Experiments are run both in a setting where ϕ is a Recurrent Neural Network (RNN), and in a setting where it is a Highway layer. In every set-up, ϕ is placed between $\theta(\mathbf{x})$ and any task-specific layer f_k. If a custom feature encoder ϕ is implemented in-between $\theta(\mathbf{x})$ and f_k, and MTL is performed, then the parameters of ϕ are shared among all $f_k(\phi(\theta(\mathbf{x})))$, where only f_k is task-specific, and therefore not shared among the full model parameter set. This is called HARD PARAMETER sharing which has proven to be more successful than SOFT PARAMETER sharing for a variety of NLP tasks [65] and is therefore considered common practice in MTL in general [16] and for NLP in particular [11, 4, 9].

Due to the fact that all three tasks are classification tasks by nature, relatedness is given on a higher, more abstract level of machine learning tasks. Although this does not guarantee relatedness with respect to MTL in particular [9], it is indispensable to stress that tasks should resemble each other not only on a lower, more task-specific but also on a higher, more task-nature related level [16]. Before I explain in more detail how ϕ is implemented, I would like to stress the importance of different task sampling strategies and give a short overview of the strategies employed in the experiments.

4.1.4 Task Sampling

To decipher whether different task sampling regimes in MTL impact model performance on the main task differently, I compare two task sampling strategies against each other in each of the implemented MTL settings. In a UNIFORM SAMPLING setting, the main task, that is QA, and the auxiliary task(s), that is subjectivity or context-domain classification, are equally often sampled during a single training epoch. Hence, in an MTL setting with one auxiliary task, each of the tasks is optimized in 50% of all training steps, and in a setting with two auxiliary tasks, a model $\tilde{\mathbf{f}}$ is fine-tuned on each of the tasks $\frac{1}{3}$ of the time.

In an OVERSAMPLING setting, however, the main task, that is QA, is sampled $\frac{2}{3}$ per training epoch, and the remaining $\frac{1}{3}$ of training steps are equally distributed among fine-tuning on the auxiliary task(s). Thus, in an MTL setting with two auxiliary tasks, each of the tasks is sampled $\frac{1}{6}$ per epoch. Contrary to the former setting, this follows a skewed sampling distribution of machine learning tasks during training, where the main task is oversampled and the auxiliary tasks are optimized equally often. The latter strategy is employed to oversample the main task and examine potential differences in performance compared to uniformly sampling all tasks.

4.1.5 Modelling Subjectivity

Before moving to the explanation of implementation details with respect to ϕ, I will briefly discuss $f_{sbj}(\theta(\mathbf{x}))$ more thoroughly. The vast majority of character sequences, namely strings, in a review paragraph $\mathbf{r}$ or more general, a context $\mathbf{c}$, consists of tokens that are not reflecting subjective opinions. Thus, it might be easier for a model $\tilde{\mathbf{f}}$ to solely classify the answer $\mathbf{a}$, which is a sub-string of $\mathbf{c}$, into subjective opinions vs. objective, measurable facts, instead of learning to predict whether an entire review reflects subjectivity.

Therefore, I will implement two different versions of $f_{sbj}(\theta(\mathbf{x}))$. In the first setting, which can be considered the standard sequence-pair classification setting for questions and contexts, $f_{sbj}(\theta(\mathbf{x}))$ is optimized to predict whether both the question $\mathbf{q}$ and its corresponding context $\mathbf{c}$ belong to the class of subjective opinions or measurable facts. In the second setting, which can be considered an exploratory modelling attempt, $f_{sbj}(\theta(\mathbf{x}))$ is trained to classify the answer $\mathbf{a}$ instead of the context $\mathbf{c}$ into subjective vs. objective. Here, one has to alternate between batches, where BATCH$_1$ consists of mini-batches of $(\mathbf{q}, \mathbf{c})$ sequence pairs and BATCH$_2$ contains mini-batches of $(\mathbf{q}, \mathbf{a})$ sequence pairs. The latter setting is called BATCH ALTERNATION as one is required to exploit sequences from BATCH$_2$ as input to $f_{sbj}(\theta(\mathbf{x}))$ and leverage sequences from BATCH$_1$ as inputs to $f_{qa}(\theta(\mathbf{x})) \vee f_{dom}(\theta(\mathbf{x}))$, of which both are generated for each training iteration.

When fine-tuning a model $\tilde{\mathbf{f}}$ on SUBJECTIVITY classification only, this is employed during both training and test time. For QA-MTL, however, the latter is deployed exclusively during training time. At inference time, where the model does not perform any auxiliary task, the model has to find an answer span $\mathbf{a}$ in a context $\mathbf{c}$ to respond to the corresponding question $\mathbf{q}$. Thus, each input sequence $\mathbf{x} = (\mathbf{q}, \mathbf{c})$.

4.1.6 Recurrent Neural Networks

Neural architectures based on the Transformer [77, 66] such as BERT do not take into account temporal dependencies between tokens in a sequence of tokens, $x_i^t \in \mathbf{x}_i^T$. Hence, I equip a model $\tilde{\mathbf{f}}$ with the possibility to exploit an RNN based neural module, namely Long-Short-Term-Memory network [35], on top of BERT prior to any task-specific linear output layer. A few recent studies have shown that further encoding the contextual feature representations yielded by BERT through LSTMs before performing the QA task enhances the learner's performance (e.g., [37]). Since BERT is the first deep neural language model (LM) based on the Transformer [21] which exploits bidirectional self-attention mechanisms, temporal dependencies must be computed forward and backward in time to retain the contextual features specific to BERT. See Section 3.3 for mathematical details of RNNs.

I refer to any RNN based POST-BERT encoding layer as ϕ, which is applied to $\theta(\mathbf{x})$, $\Rightarrow \phi(\theta(x))$. I leverage PyTorch's LSTM implementation [54]. For each recurrent POST-BERT encoder, 2 LSTM layers are deployed, the bidirectional flag of the LSTM class is set to true to employ a BiLSTM as discussed in Section 3.3. Additionally, a dropout rate of .25 is applied to each layer. I call this an LSTM block. If a model is equipped with a POST-BERT recurrent encoding layer, then it consists of one and only one LSTM block.

4.1.7 Highway Networks

Another way of further encoding BERT's feature representations is to pass a word sequence's latent representation $\mathbf{z}_i$ through a Highway network [73]. Again, the same study as mentioned in the previous section [37] showed that a Highway layer in-between BERT, that is $\phi(\theta(x))$ or $\phi(z)$, and a task-specific output layer, that is f_k, supports the information flow between BERT and the linear QA output layer, thus enhancing model performance. See Section 3.4 for a thorough introduction about and mathematical details of Highway networks.

I refer to any POST-BERT encoding layer that leverages a Highway block as ϕ, which is applied to $\theta(\mathbf{x})$, $\Rightarrow \phi(\theta(x))$. As such, ϕ may be employed as an RNN module or a Highway block, depending on the set-up. The potential caveats for a recurrent POST-BERT encoding mentioned in Section 3.3 do not hold for Highway blocks since a Highway network, similar to BERT, does not leverage recurrence, thus making this set-up potentially better suited for STL classification tasks. If a model is equipped with a POST-BERT Highway layer, then it consists of one and only one Highway block. This is done to not overload the computational budget and keep the number of parameters in a model $\tilde{\mathbf{f}}$ similar across set-ups.

4.2 Fine-tuning

Each pre-trained model $\tilde{\mathbf{f}}$ is fine-tuned either on D_s, D_o or D_c. D_s refers to the dataset that only consists of question-review pairs from SUBJQA, D_o denotes the dataset that exclusively contains question-context pairs from SQUAD, and $D_c = D_s \cup D_o$. $\forall \tilde{D} \in \{D_s, D_o, D_c\}$ fine-tuning is performed for a predefined number of $T = 3$ epochs. This follows the fine-tuning regime as recommended in the original BERT paper [21]. During each epoch t, a model $\tilde{\mathbf{f}}$ updates its weights for $\frac{N^t}{b}$ steps, where N^t refers to the number of examples in a given train set X^t and b denotes the batch size. The latter is set to 16 for all training procedures, which alongside batch sizes of 32 and 64 is considered standard practice [21, 68]. I chose $b = 16$ since in initial experiments mini-batches of size $b = 32$ did not fit into GPU memory for a Titan X with 12 GB given my training procedure and model set-up.

4.2.1 Evaluation

To inspect whether a model $\tilde{\mathbf{f}}$, that is exposed to the training set X^t, generalizes well to unseen validation data X^v, and does not overfit to X^t, a researcher is required to evaluate $\tilde{\mathbf{f}}$ on X^v a predefined number of k times during training. The most common set-up for performing this step is to either test $\tilde{\mathbf{f}}$ exactly once after the entire training ($k = 1$) or after an epoch t ($k = T$) [21, 58]. However, one recent study has examined this evaluation regime in more detail [23]. The authors showed that evaluating $\tilde{\mathbf{f}}$ on X^v a predefined number of 10 times during an epoch ($k = T \times 10$) leads to significantly better performance than exploiting either of the aforementioned standard evaluation set-ups. Thus, I implement both a set-up where $\tilde{\mathbf{f}}$ is evaluated after a training epoch t ($k = T$), and an alternative version where $\tilde{\mathbf{f}}$ is tested 10 times on X^v during an epoch t ($k = T \times 10$).

For various reasons, I do not implement the third evaluation set-up, where $\tilde{\mathbf{f}}$ is evaluated once on X^v after the entire training procedure ($k = 1$). Firstly, this is less common practice in machine learning research than evaluating $\tilde{\mathbf{f}}$ after each epoch t. Secondly, it seems less reasonable to implement a $k = 1$ regime, if one endeavours to stop the training procedure early when $\tilde{\mathbf{f}}$ either decreases or plateaus with respect to its performance. This performance, however, cannot be measured on X^t since it is not a valid indicator for generalization. The model is trained on X^t and hence has already seen all examples from X^t as many items as was iterated over X^t. A researcher must therefore investigate whether performance drops or reaches a plateau with respect to X^v to correctly implement an early stopping regime. One can see that the latter cannot be done, if $k = 1$ - at least not in the standard way of early stopping the training.

4.2.2 Early Stopping

I implement early stopping for two reasons. Firstly, it is computationally inefficient to train a model $\tilde{\mathbf{f}}$ a total number of T epochs on X^t, if model performance on X^v does not increase or even plummets towards the end of training. To save time and computational budget, one could simply terminate the training when this happens. Secondly, one wants to save the weights of $\tilde{\mathbf{f}}$ at its peak performance on X^v during training, and not after a performance drop. Early stopping during training is performed, whenever the cross-entropy loss with respect to QA evaluated on the entire validation set $X^v = \{(x_j^v, y_j^v)\}_j^{n_s}$ does not increase for $k = 5$ evaluation steps, if $k = T \times 10$, or is higher than the loss at the $(k - 1)^{th}$ evaluation step, if $k = T$, since $T = 3$ for all fine-tuning regimes, and hence $(k-2)^{th}$ at $k = 3$ for $k = T$ is validation performance after epoch 1, which we do not want to compare against.

The latter early stopping regimes are executed after a model $\tilde{\mathbf{f}}$ is fine-tuned for a single training epoch since I do not want to stop training during the first epoch. The weights of a model $\tilde{\mathbf{f}}$ are saved only when the validation loss at the k^{th} evaluation step is lower than the previous minimum loss, and not stored when an increase in validation loss is observed.

4.2.3 Optimization

Each model $\tilde{\mathbf{f}}$ is optimized through Adam [40] with weight decay fix as recommended in [21]. The learning rate η is set to $5e-5$ $\forall\tilde{\mathbf{f}}$ as recommended for fine-tuning BERT on QA. Furthermore, a linear scheduler with a warm-up period is applied to the optimizer to control η during training [36]. This works as follows: η linearly increases for a predefined number of steps, called WARM-UP STEPS, and after the warm-up period decreases linearly until model convergence or stopping of the training procedure.

For the simplest training set-up, that is a single task learning (STL) setting where a model $\tilde{\mathbf{f}}$ performs the main task (i.e., QA) only, the empirical risk is minimized per iteration through mini-batch gradient descent optimization as follows,

$$\min_{\Theta} \frac{1}{2} \left(\frac{1}{n} \sum_{i=1}^{n} J\left(f_{qa}\left(\phi\left(\theta\left(\mathbf{x}_i\right)\right)\right)^s, y_i^s\right) + \frac{1}{n} \sum_{i=1}^{n} J\left(f_{qa}\left(\phi\left(\theta\left(\mathbf{x}_i\right)\right)\right)^e, y_i^e\right) \right)^3, \tag{4.1}$$

where $f_{qa}(\phi(\theta(x_{(i)})))$ is the conditional probability that the QA model knows that the question-context sequences $x_{(i)}$ within a mini-batch of n sequences correspond to the the correct start and end positions $y_{(i)}^s$ and $y_{(i)}^e$ of the respective answer spans $y_{(i)}^{s-e}$, and $J(\theta)$ is the cross-entropy loss function with respect to all model parameters θ which is computed for both $y_{(i)}^s$ and $y_{(i)}^e$ as follows,

[3] The parameter set ϕ of the custom encoder structure is optional, but for simplicity depicted in the equation. This holds for other equations too.

$$J(\Theta) = -\sum_{y_i}^{n} 1(X, y_i) \log(P_r(y_i | f_{qa}(\phi(\theta(x_i))))), \tag{4.2}$$

where $1(x_i, y_i)$ is the binary indicator function (0 or 1) if the start or end position respectively is correct for the question-context sequence x_i, and $P_r(y_i | f_{qa}(\phi(\theta(x_i)))$ is a given discrete probability distribution over all possible start and end positions respectively. The latter is computed by the SOFTMAX function as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_j}}, \tag{4.3}$$

where $(\mathbf{z})_i$ evaluates to $f_{qa}(\phi(\theta(x_i)))$ and denotes the non-normalized output of an NN model $\tilde{\mathbf{f}}$, in the literature referred to as LOGITS. Logits are k-dimensional vectors, where k denotes the possible number of classes (i.e., start and end positions respectively). For QA, this is an output matrix of size $k \times 2$. Hence, $\mathbf{z}_i \in \mathbf{R}^{k \times 2}$. Following the standard practice we compute start and end logits, however, separately and split the resulting raw output $\mathbf{z}_i \in \mathbf{R}^{k \times 2}$ into $z_i^s \in \mathbf{R}^k$ and $z_i^e \in \mathbf{R}^k$ respectively to compute the SOFTMAX over z_i^s and z_i^e to yield probability distributions p_i^s and p_i^e. Since the SOFTMAX function is nothing other than a multinomial logistic regression over k classes, we can simply write σ to denote SOFTMAX. Thus, we must compute $\sigma(z^s)_i$ and $\sigma(z^s)_i$ separately.

4.3 Multi-task Learning

In MTL, we minimize the following empirical risk for multi-way classification of CONTEXT-DOMAINS d or binary classification of SUBJECTIVITY labels of questions and answers respectively,

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} J\left(f_{aux}\left(\phi\left(\theta\left(\mathbf{x}_i\right)\right)\right)^{aux}, y_i^{aux}\right), \tag{4.4}$$

where J denotes the categorical cross-entropy as described in Equation 4.2 for multi-way classification and the binary cross-entropy for binary classification, N refers to the total number of mini-batches and $f_{qa}(\phi(\theta(x_{(i)})))^{aux}$ is the conditional probability that the model classified the input sequence $x_{(i)}$ in a mini-batch of n examples, where $n = 16$, as the corresponding true domains $y_{(i)}^{d}$ or the correct SUBJECTIVITY labels $y_{(i)}^{sbj}$. Whenever label imbalance is observed, loss weighting is performed. That is, the empirical risk for classes that appear less frequently in the training data is weighted higher. To provide an example, imagine that positive examples account for 100 of all training examples N, where $N = 900$, in a binary classification problem with a single class. Then, the loss for positive examples is multiplied by a factor of $\frac{800}{100} = 8$ such that the loss acts as if the dataset contains equally many positive and negative examples. Similarly, to account for label imbalance in the multi-class problem cross-entropy loss weights corresponding to each class are computed as follows,

$$w^k = 1 - \frac{n^k}{N}, k = 1, \cdots, \mathbf{k} \tag{4.5}$$

Hence, cross-entropy loss for each class is weighted according to the class weights obtained from Computation 4.5. For multi-way classification, the SOFTMAX function as depicted in Equation 4.3 is computed over the model's raw output logits to yield a discrete probability distribution over all possible domains k. In binary classification, however, the SIGMOID function is computed over the model's raw output logits to obtain a single scalar value between $0 - 1$ (i.e., probability value), according to the following formula,

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{e^{z_i} + 1} \tag{4.6}$$

where $(\mathbf{z})_i$ evaluates to $f_{sbj}(\phi(\theta(x_i)))$ and denotes the model's SUBJECTIVITY label prediction.

4.4 Adversarial Training in MTL

To inspect whether models benefit from learning domain invariant features, which has proven to be useful in various machine learning problems where labels from the target domain were limited [19, 30, 27, 49, 1], I implement two different adversarial training settings in MTL. In the first adversarial training setting, which I will refer to as ADVERSARIAL SIMPLE throughout the following section, the loss is simply reversed after each comparison between the model's raw output logits $f_{qa}(\phi(\theta(x_{(i)})))$ and the true labels $y_{(i)}^{aux}$.

In the second adversarial training setting, which I refer to as ADVERSARIAL GRL throughout the following section, a gradient reversal layer (GRL) following [27], is placed in-between the shared feature encoding layers $\phi(\theta(\mathbf{x}))$ and the auxiliary task-specific output layers f_{aux}.

4.4.1 Reversing Losses

In an ADVERSARIAL SIMPLE training setting, the sign of the loss is simply reversed to make the model not learn the auxiliary task(s) at all, whereas all other optimization parameters stay as in a NORMAL MTL setting. Hence, goal of the optimization procedure is to maximize the loss for auxiliary task(s):

$$\max_{\Theta} \frac{1}{N} \sum_{i=1}^{N} J\left(f_{aux}\left(\phi\left(\theta\left(\mathbf{x}_i\right)\right)\right)^{aux}, y_i^{aux}\right), \tag{4.7}$$

where J denotes the categorical or binary cross-entropy as described in Equation 4.2, N refers to the total number of mini-batches and $f_{aux}(\phi(\theta(x_{(i)})))$ is the conditional probability that the model classified an input sequence $x_{(i)}$ in a mini-batch of n examples, where $n = 16$, according to the goal of the particular auxiliary task.

4.4.2 Reversing Gradients

Following Ganin et al., 2014 [27], a Gradient Reversal Layer (GRL) is placed in-between the shared feature encoding layers $\phi(\theta(\mathbf{x}))$ and the auxiliary task-specific output layers f_{aux}. The primary goal of ADVERSARIAL GRL is to produce domain-invariant feature representations while at the same time making the model learn the auxiliary task(s). This is opposed to AUXILIARY SIMPLE where the models are optimized in way that does not make them learn the auxiliary tasks at all. Hence, the optimization follows a normal training setting, as depicted in Equation 4.4. In contrast to a normal training setting, here the gradients for the feature extractor(s), $\phi(\theta(x))$, are reversed. Thus, the partial derivative $\frac{\partial L_{(d)}}{\partial \theta_{(f)}}$ is scaled by $-\lambda$, which was set to 1 for all experiments, such that the backpropagated gradient is simply reversed and therefore negative. See Figure 4.2 for an overview of the GRL model architecture and Figure 4.1 for how gradients are reversed in my work.

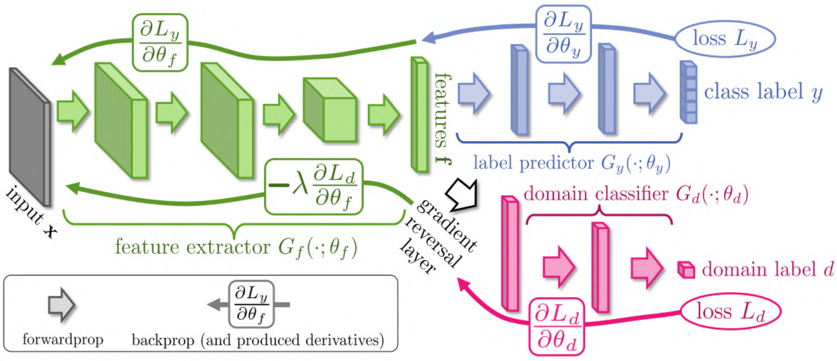

$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
input x
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 4.2: GRL architecture [27].

4.5 Sequential Transfer

To investigate a model that is trained on both auxiliary tasks and the main task sequentially, I fine-tune BERT in its simplest set-up, that is without any additional custom encoding layer ϕ, on all three tasks, namely context-domain classification, subjectivity classification and QA, sequentially. This can also be referred to as SOFT PARAMETER sharing, where each task has both a separate feature encoder and task-specific output layer but information concerning the model's hidden representations is shared across tasks to ultimately help the model enhance performance on the main task.

To inspect whether information from the auxiliary tasks is useful to perform better on the main task, the model is evaluated on the train set in an additional epoch after convergence on each of the two auxiliary tasks. During this additional synthetic evaluation epoch, the model's raw output logits are stored for each input sequence $x_i \in D^t$. To yield smoother distributions and obtain actual probability scores, the logits are passed through a SIGMOID (see Equation 4.6) and a SOFTMAX (see Equation 4.3) function for subjectivity and context-domain classification respectively. I refer to these vectors of concatenated probability scores as SOFT TARGETS. During QA, the soft targets corresponding to both subjectivity and context-domain classification, $\mathbf{p}_i \in \mathbf{R}^K$, are concatenated with the matrix of hidden representations for each input sequence, $\mathbf{H}_i^l \in \mathbf{R}^{T \times D}$, at the last transformer layer before performing the classification, yielding a new matrix of hidden representations, $\mathbf{H}_i^l \in \mathbf{R}^{T \times (D+K)}$.[4]

In addition, I implement another set-up whose computations scarcely deviate from the aforementioned setting but leverage hard instead of soft targets in the concatenation part. I refer to this setting as ORACLE. The vector $\mathbf{p}_i \in \mathbf{R}^K$ consists of two parts, namely $\mathbf{p}_i^s \in \mathbf{R}^{K_s}$ and $\mathbf{p}_i^d \in \mathbf{R}^{K_d}$.[5] Whereas $\mathbf{p}_i^s \in \mathbf{R}^{K_s}$ contains two 1s, iff both the answer and the question are subjective, a single 1 and one 0, iff the answer or the question is subjective, and two 0s, iff both are objective, $\mathbf{p}_i^d \in \mathbf{R}^{K_d}$ evaluates to a one-hot-encoded vector with a single 1 at the index of the correct domain and 0s otherwise. The concatenation with the matrix of hidden representations, $\mathbf{H}_i^l \in \mathbf{R}^{T \times D}$, is the same as above, thus yielding $\mathbf{H}_i^l \in \mathbf{R}^{T \times (D+K)}$.

In this set-up, no adversarial training is performed as the model is meant to learn each task separately. The architecture and the corresponding fine-tuning procedure is illustrated below in Figure 4.3.

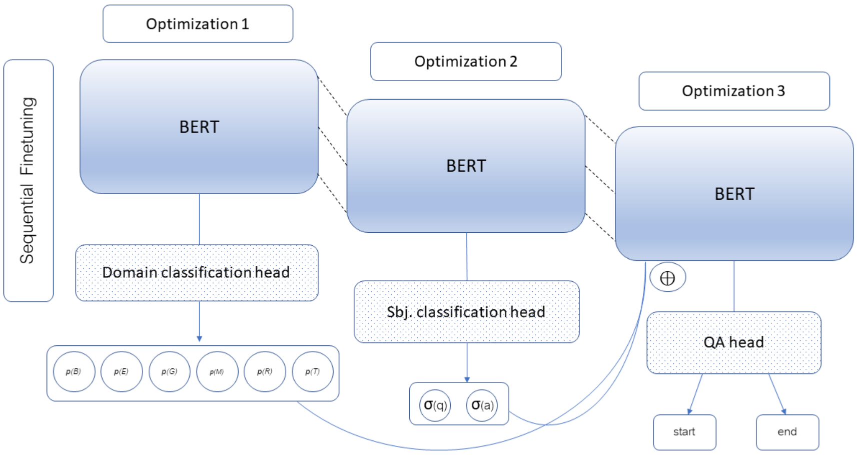


Figure 4.3: Sequential transfer. BERT is sequentially fine-tuned on each task in the task set $\mathbf{T}$. In so doing, the model transfers knowledge from the previous tasks to the current task through SOFT PARAMETER sharing, and ultimately injects information from the auxiliary tasks into BERT for QA through the concatenation of soft (auxiliary) targets with BERT's hidden representations for each token w.r.t. an input sequence x_i. The concatenation happens at the last Transformer layer only.

[4] $K = 8, T = 512, D = 768, l = 6$.

[5] $K_s = 2, K_d = 6$.

Chapter 5

Data

5.1 SQuAD

QA TYPE	QUESTION	ANSWER
OBJECTIVE	"How many awards did Beyoncé win at the 46th Grammy's Awards?"	"five"
OBJECTIVE	"How many nights did Beyoncé play at the resort?"	"four"
OBJECTIVE	"What instrument did Auguste Franchomme play?"	"cello"
OBJECTIVE	"Many reviewers consider the second part of the book to be about what issue?"	"race relations"

Table 5.1: Examples of answerable questions and their corresponding answers in SQuAD.

At the time of writing, SQuAD is the most popular and largest span-based QA data set to train and evaluate machine reading systems on [61, 21]. There are two versions of SQuAD, namely SQuAD v1.0 [62] and SQuAD v2.0 [61]. Owing to the fact that SQuAD v2.0 is both the latest and more complex span-based QA data set of the two as well as more similar to SubjQA than SQuAD v1.0, I decided on exploiting SQuAD v2.0 only to compare against SubjQA. The main difference between SQuAD v1.0 and SQuAD v2.0 is that SQuAD v2.0 contains questions that are not answerable given the corresponding paragraph. Moreover, question-paragraph pair sequences are slightly longer in SQuAD v2.0 than in SQuAD v1.0 [61]. This makes it similar to the nature of SubjQA which even consists of more unanswerable questions and longer context sequences than SQuAD v2.0 (see Table 5.2). For simplicity and to avoid numbering, I will refer to SQuAD v2.0 with SQuAD throughout the following sections.

As depicted in Table 5.1, answerable questions in SQuAD have a clear and objective answer whose string span is part of the corresponding Wikipedia paragraph. Questions and answers in SQuAD are both extracted from various Wikipedia articles, and thus contribute to highly accurate English grammar as most Wikipedia articles usually undergo proof-reading through independent reviewers. This is in stark contrast to the subjective QA dataset SubjQA as I will discuss in the following section.

SOURCE \ SPLIT	TRAIN				DEV	TEST
	n questions	% answerable	% objective	% subjective	n questions	n questions
SQuAD	15,228	53.5	100.0	0.0	3,807	_
SubjQA	14,630	44.0	17.3	82.7	1,595	4,075

Table 5.2: Overview of SQuAD and SubjQA across dataset splits.

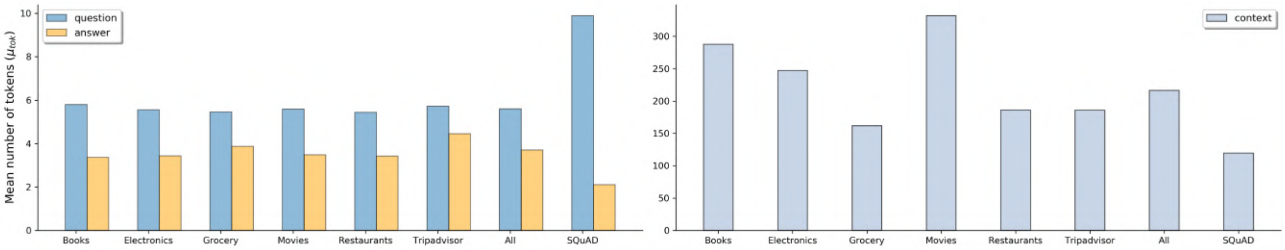


Figure 5.1: Average number of tokens (μ) per word sequence x_i across all domains in SubjQA and SQuAD of which the latter was treated as its own domain. The bar graph on the left-hand side depicts the average number of tokens for questions and answers respectively. The right bar chart shows the average number of tokens for reviews (SubjQA) and Wikipedia paragraphs (SQuAD) respectively.

5.2 SubjQA

DOMAIN \ SOURCE	SQUAD		SUBJQA		
	Train	Dev	Train	Dev	Test
	n examples	n examples	n examples	n examples	n examples
books	_	_	2,503	264	573
electronics	_	_	2,382	267	675
grocery	_	_	2,827	313	322
movies	_	_	2,456	273	632
restaurants	_	_	2,349	231	799
tripadvisor	_	_	2,113	247	1,074
wikipedia	15,228	3,807	_	_	_

Table 5.3: Distribution of paragraph or review domains across dataset splits in SQuAD and SubjQA respectively.

SubjQA is a recently developed span-based QA data set that contains both objective and subjective questions [13]. Due to the fact that the dataset is meant to be subjective in nature the latter set of questions is with 82.7% of the total number of questions highly overrepresented (see Table 5.2). This makes it particularly difficult for QA models as they are required to learn about and understand the subjective features of a question q_i. Similarly to SQuAD, about half of the questions ($\sim 56.0\%$) are not answerable given the corresponding context. This is another detail of the dataset that makes it more difficult than other datasets that exclusively contain answerable questions. In contrast to SQuAD, SubjQA does not target common knowledge (e.g., "What is the birth place of Barack Obama?") which is likely to have occurred in the data used for pre-training of deep LMs such as BERT [21, 25]. The lack of targeting common knowledge, frequently contained in encyclopedias such as Wikipedia, enhances the difficulty of answering questions w.r.t. this dataset and makes fine-tuning indispensable.

In SubjQA, the context corresponding to a question q_i is a review paragraph r_i belonging to one of six domains, where $r_i \in \{$BOOKS,ELECTRONICS, GROCERY, MOVIES, RESTAURANTS, TRIPADVISOR$\}$. A review r_i never belongs to more than a single domain. As depicted in Table 5.3, the data set is fairly balanced with respect to the different review domains. This is different to SQuAD, which does only consist of paragraphs extracted from Wikipedia (single domain). Hence, when fine-tuning on SubjQA it is crucial to inform a QA model about linguistic domain variances and shifts to not end up with an architecture that performs well on one or few domains but poorly on the rest.

What's compelling about the data set is that question-answer types (i.e., subjective vs. objective) were manually annotated by human workers. This ensures the reliability of the labels. Crowdworkers were asked whether the respective question is asking about the subjective opinion of the reviewer or about an objective,

QA TYPE	QUESTION	ANSWER
OBJECTIVE	"How is the read?"	"I thoroughly enjoyed reading about America"
SUBJECTIVE	"Do you think the audio is very strong?"	"the sound is decent"
OBJECTIVE	"How good is the camera of the nook?"	"the camera is excellent"
SUBJECTIVE	"Do you want some tea?"	"I drink Lipton iced tea"
OBJECTIVE	"Which flavor was there?"	"salad"
SUBJECTIVE	"What is the quality of the product?"	"I didn't think it was bad"
OBJECTIVE	"How helpful is the front desk?"	"staff were pleasant and helpful"
SUBJECTIVE	"How good are the actors in this film?"	"the actors are brilliant"

Table 5.4: Examples of answerable questions and their corresponding answers in SubjQA.

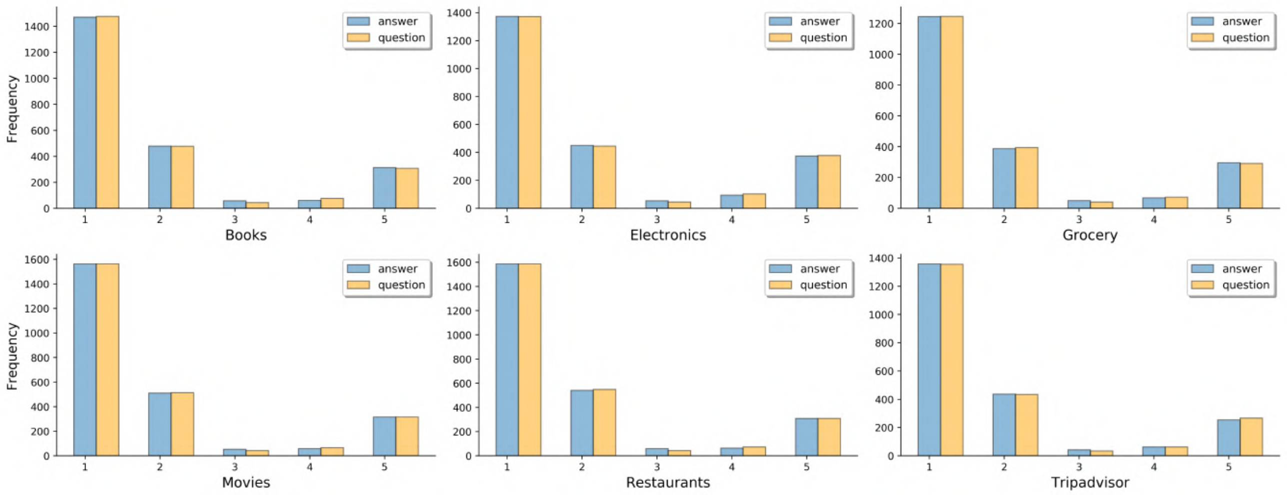


Figure 5.2: Frequencies of objectivity vs. subjectivity levels assigned to questions and answers respectively by human crowdworkers across the different domains in SubjQA. Possible objectivity vs. subjectivity levels were integer values between 1 - 5 on a classic Likert scale. The lower the assigned value (< 3), the more likely the question or answer expressed a subjective opinion opposed to an objective, measurable fact.

SQUAD	SUBJQA	
OBJECTIVE	OBJECTIVE	SUBJECTIVE
what: 43.97%	how: 42.32%	how: 61.41%
who: 9.92%	what: 23.80%	what: 16.53%
how: 8.20%	is: 10.49%	is: 7.95%
when: 5.30%	where: 6.62%	does: 2.93%
in: 4.77%	does: 4.68%	do: 2.23%

Table 5.5: Top 5 interrogative words for the train datasets of SQuAD and SubjQA respectively. For this purpose, SubjQA was split into objective and subjective questions according to the annotations human crowdworkers provided (see Figure 5.2). Percentage to the right of each token denotes % of questions that started with this prefix in the respective dataset split.

measurable fact. Moreover, they were asked whether the selected answer span - crowdworkers had to select the correct answer span in the review - expresses a subjective opinion or an objective measurable fact. In so doing, workers had to to assign an integer value between $1 - 5$ according to a Likert scale [3] to both a question and the provided answer (see Figure 5.2). The higher the assigned value, the more objective the question or answer respectively appeared to the crowdworker.

What can be inferred from Table 5.5 is the fact the distributions of prefixes of questions between objective and subjective questions in the train set of D_{SubjQA} do not notably differ from one another. It seems as if the vast majority of questions starts with the same prefix no matter whether the question was labelled objective or subjective by the human crowdworkers. The difference in the prefix distribution between D_{SubjQA} and D_{SQuAD}, however, appears to be more apparent (see Table 5.5). This might reflect a potential caveat for the task of classifying a "question - context" $(\mathbf{q}, \mathbf{c})$ pair sequence into subjective vs. objective when exploiting D_{SubjQA} only.

Chapter 6

Quantitative Analyses

6.1 Question Answering

6.1.1 Single-task Learning

MODEL \ FINE-TUNING	SUBJQA		COMBINED	
	Exact-match	$F1$	Exact-match	$F1$
BERT	76.04	76.49	**75.37**	**76.13**
BERT + Highway	75.95	76.57	73.86	75.37
BERT + BiLSTM	**76.06**	**76.93**	74.63	75.79
$\bar{\Theta}_{\mathbf{QA}}$	76.02	76.66	74.62	75.76

Table 6.1: Single Task Learning (STL) - Question Answering (QA). Models were either fine-tuned on SUBJQA or both SQUAD and SUBJQA which I refer to as COMBINED, and evaluated on SUBJQA only. Each model consisted of a pre-trained DISTILBERT feature extractor, custom POST-BERT encoding layers and a task-specific (QA) output layer that were all jointly fine-tuned on either of the two $D_{i\,\in\,\{subj,\,comb\}}$ versions. Best results are depicted in bold face.

In a STL setting, all implemented models were fine-tuned on $D_{i\,\in\,\{subj,\,comb\}}$ to exclusively perform QA. Each model consisted of a pre-trained DISTILBERT BASE feature extractor, custom POST-BERT encoding layers (see Section 4 for further details) and a task-specific output layer for QA that were all jointly fine-tuned on either of the two $D_{i\,\in\,\{subj,\,comb\}}$ versions. Inference was performed on the test set of D_{subj} only to evaluate which fine-tuning regime yields better model performance in respect of SubjQA.

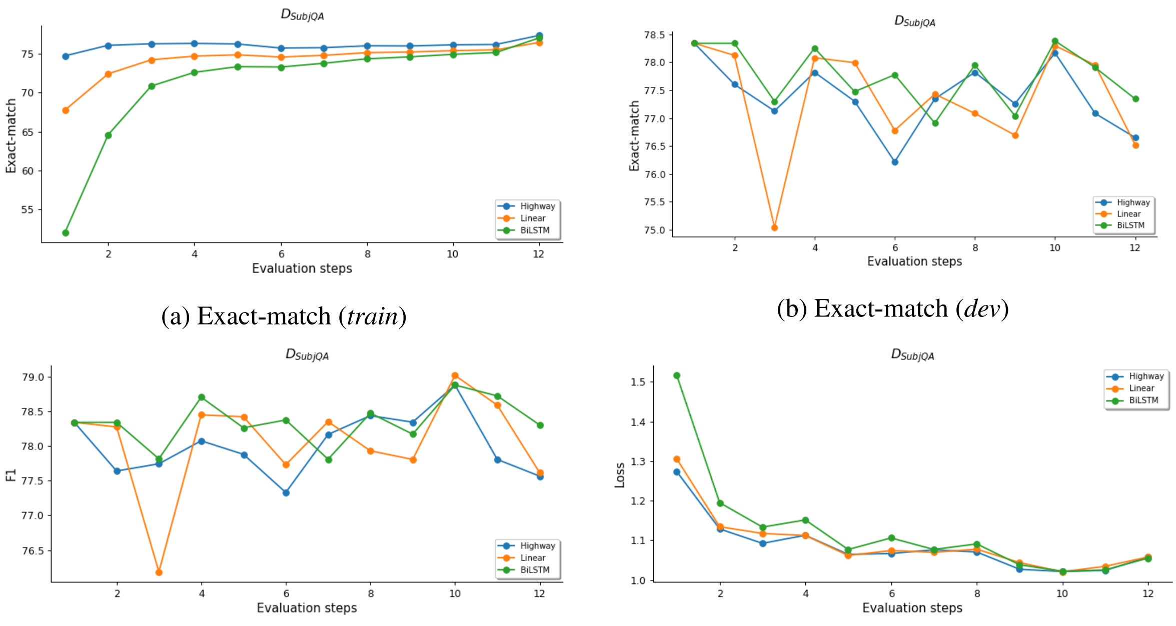
To validate the necessity of fine-tuning the models on $D_{i\,\in\,\{subj,\,comb\}}$ before performing inference on D_{subj}, I tested a DISTILBERT BASE model that was previously fine-tuned on SQuAD.[1] SQuAD is a span-selection QA dataset that consists of only objective questions from a single domain (i.e., WIKIPEDIA) [62]. Hence, a model that was fine-tuned exclusively on SQuAD is likely to not perform well on D_{subj}. As expected, the performance of the pre-trained DISTILBERT BASE model was with an exact-match and an $F1$-score of 34.59% and 39.66% rather moderate. The higher an $F1$-score, the better did a learner perform. To create conditions that allow for fair comparisons between models, I fine-tuned my DISTILBERT BASE implementation on 80% of the official SQuAD train set, and evaluated it on the test set of SubjQA. This model yielded an exact-match accuracy and an $F1$-score of 59.58% and 61.70% respectively, which is significantly higher than the publicly available pre-trained version. This is most likely due to the fact that the publicly available model was fine-tuned on SQuAD v1.0, whereas I have exploited SQuAD v2.0 for all experiments. This model served as the BASELINE model. Hence, the performances of all subsequently implemented models were compared against its exact-match and

[1] https://huggingface.co/transformers/model_doc/distilbert.html#distilbertforquestionanswering

$F1$-scores with respect to D_{subj}. This experiment was conducted to inspect **Research Question (RQ)** 1. Recall that **RQ** 1 aimed at investigating whether it is necessary to fine-tune BERT on SubjQA for achieving a high score or whether it is sufficient to use a model fine-tuned on SQuAD.

Fine-tuning a model on D_{comb} yielded worse performance than fine-tuning a model on D_{sbj} as shown in Table 6.1. The results are, however, not considerably different. This indicates that fine-tuning exclusively on SubjQA appears crucial to achieve the highest possible performance. However, fine-tuning a model on D_{comb} might let a learner perform well with respect to both SubjQA and SQuAD without a significant deterioration in performance compared to models trained solely on the task-specific datasets.

The following analyses were performed to examine **RQ** 2, which sought insight about the benefits of additional LSTM or Highway layers on top of BERT with respect to downstream performance. Indeed, additional POST-BERT encoding layers, namely a Highway network or a BiLSTM, both increased $F1$-scores. The best STL model, BERT$_{\mathbf{QA}}$ + BiLSTM, achieved an $F1$-score of 76.93% which is a relative improvement of .58% over BERT$_{\mathbf{QA}}$ which scored 76.49% $F1$. The improvement of models with additional POST-BERT encoding layers, ϕ, is also reflected in the $F1$-scores and exact-match accuracies with respect to the development set as a function of evaluation steps (see Figure 6.1). Performance on the validation set varied notably less for models with an additional Highway layer or a BiLSTM in-between BERT and the linear QA output layer. Fewer fluctuations are in general an indicator of both more stable learning and less randomness involved in the performances.



(a) Exact-match (*train*)

(b) Exact-match (*dev*)

(c) $F1$ (*dev*)

(d) Evaluation loss (*dev*)

Figure 6.1: STL training w.r.t. Question Answering (QA). Models were fine-tuned and evaluated on SUBJQA. Depicted are exact-match accuracies, $F1$-scores and cross-entropy losses as a function of evaluation steps for both train and development sets of D_{sbj} across all implemented STL QA models.

6.1.2 Multi-task Learning

Experiments in this subsection were performed to investigate into **RQ** 3. The goal of **RQ** 3 was to scrutinize whether MTL and adversarial training regimes improve upon single-task learning.

6.1.3 Parallel Transfer: QA and Subjectivity classification

In the current MTL setting, mini-batches were alternated between all tasks in the task set $\mathbf{T} = \{T, T'\}$, namely QA (main), and subjectivity classification (auxiliary). In a UNIFORM SAMPLING setting, a model $\tilde{\mathbf{f}}$ was fine-tuned on each of the tasks $\frac{1}{2}$ of the time according to the strategy of uniformly sampling tasks outlined in Section 4.1.4. In an OVERSAMPLING setting, however, QA was sampled $\frac{2}{3}$ per epoch, and the remaining $\frac{1}{3}$ of training steps were allocated to subjectivity classification (see Section 4.1.4 for further details).

As can be inferred from Table 6.2, on average OVERSAMPLING yielded better results than UNIFORM SAMPLING. The difference between the mean performances was statistically significant at $\alpha = .05$ according to an independent *t*-test. Hence, oversampling the main task considerably increased performance over uniformly sampling both tasks. Additional POST-BERT recurrent encoding layers (i.e., BiLSTMs) did not enhance performance over set-ups without such additional recurrent layers. This holds for the UNIFORM SAMPLING train version. Although slight improvements can be reported for an OVERSAMPLING set-up, there is no statistical difference between the two implementations. In MTL, I did not perform experiments where learners leveraged (shared) Highway layers to keep a reasonable number of experiments and not exceed the constrained computational budget. In an OVERSAMPLING setting, learners that were trained adversarially with respect to the auxiliary task improved over models that were not.

The overall best model was a learner that was trained adversarially on subjectivity classification with respect to $(\mathbf{q}, \mathbf{a})$ input sequences, namely BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple), with an observed exact-match accuracy of 76.56% and an $F1$-score of 76.94%. This was the highest exact-match accuracy across training set-ups and model implementations. One other model, however, performed slightly better with respect to $F1$. This was another adversarially trained model, namely BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM + adversarial (GRL), which leveraged a POST-BERT shared recurrent encoding layer and a gradient reversal layer (GRL) with respect to the auxiliary task. It achieved an $F1$-score of 76.98% which was the highest reported $F1$-score overall.

Model \ Fine-tuning	SubjQA		Combined	
	Exact-match	$F1$	Exact-match	$F1$
Auxiliary $\mathbf{1}$ - uniform sampling				
BERT$_{\mathbf{QA+Sbj(q,c)}}$	76.24	76.94	74.29	75.69
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + adversarial (simple)	76.03	76.33	72.91	74.64
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + adversarial (GRL)	75.69	76.60	76.36	76.57
BERT$_{\mathbf{QA+Sbj(q,a)}}$	76.26	76.29	73.54	74.75
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple)	75.67	76.43	72.97	74.20
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (GRL)	76.26	76.47	72.85	74.34
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM	76.30	76.40	74.55	75.68
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM + adversarial (simple)	75.34	76.31	74.13	75.43
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM + adversarial (GRL)	74.57	75.83	74.15	75.45
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM	**76.38**	76.43	73.34	74.72
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM + adversarial (simple)	75.91	76.76	73.34	74.72
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM + adversarial (GRL)	75.91	76.76	74.78	75.87
$\bar{\Theta}_{\mathbf{QA+Sbj(q,\,a \vee c)}}$	75.88	76.46	73.93	75.17
Auxiliary $\mathbf{1}$ - oversampling				
BERT$_{\mathbf{QA+Sbj(q,c)}}$	75.95	76.23	74.17	75.34
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + adversarial (simple)	76.18	76.58	75.22	75.59
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + adversarial (GRL)	76.22	76.55	76.14	**76.65**
BERT$_{\mathbf{QA+Sbj(q,a)}}$	76.14	76.54	73.76	75.15
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple)	**76.56**	76.94	75.91	76.51
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (GRL)	76.03	76.32	72.99	74.63
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM	76.18	76.90	75.97	76.44
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM + adversarial (simple)	76.36	76.58	**76.38**	76.60
BERT$_{\mathbf{QA+Sbj(q,c)}}$ + BiLSTM + adversarial (GRL)	76.26	**76.98**	75.97	76.44
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM	76.12	76.93	74.82	76.21
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM + adversarial (simple)	76.18	76.86	73.62	74.98
BERT$_{\mathbf{QA+Sbj(q,a)}}$ + BiLSTM + adversarial (GRL)	75.97	76.87	74.21	75.52
$\bar{\Theta}_{\mathbf{QA+Sbj(q,\,a \vee c)}}$	76.18	76.69 *	75.00 *	75.95 *

Table 6.2: Multi-task learning (MTL) - Question Answering (QA) with one auxiliary task, namely subjectivity classification. In a uniform sampling task setting, all tasks - main and auxiliary task - were randomly sampled according to a uniform distribution, whereas in an oversampling setting, the main task (i.e, QA) was sampled $\frac{2}{3}$ per epoch. Models were either fine-tuned on SubjQA or both SQuAD and SubjQA which we call combined, and evaluated on SubjQA only. Each model consisted of a shared pre-trained distilbert feature extractor, optional shared post-BERT recurrent encoding layers (i.e., BiLSTMs) and task-specific output layers that were jointly fine-tuned on either of the two $D_{i \,\in\, \{subj,\, comb\}}$ versions. Adversarial simple refers to adversarial training were the sign of the loss was simply reversed to make the model not learn the auxiliary task at all. Adversarial GRL refers to a more sophisticated adversarial strategy, namely a Gradient Reversal Layer (GRL) between the shared encoding layers and the task-specific output layers. * indicates a statistically significant difference between oversampling and uniform sampling according to an independent t-test at $\alpha = .05$.

6.1.4 Parallel Transfer: QA, Subjectivity and Context-domain classification

In the following MTL setting, mini-batches were alternated between all tasks in the task set $\mathbf{T} = \{T, T', T''\}$, namely QA, subjectivity classification, and context-domain classification.

When fine-tuning a model $\tilde{\mathbf{f}}$ on D_{Subj}, the learner is required to find a correct answer span a_i that either reflects a subjective opinion or an objective, measurable fact. Thus, subjectivity classification (AUX$_1$) appears to be a useful auxiliary task, as we have seen in the section above. Moreover, the answer span must be extracted from a review that belongs to different linguistic domains. Hence, context-domain classification (AUX$_2$) might help a learner to better understand the review it must find an answer span a in.

In a UNIFORM SAMPLING setting, a model $\tilde{\mathbf{f}}$ was fine-tuned on each of the tasks $\frac{1}{3}$ of the time according to the strategy of uniformly sampling tasks outlined in Section 4.1.4. In an OVERSAMPLING setting, however, QA was sampled $\frac{2}{3}$ per epoch, and the remaining $\frac{1}{3}$ of training steps were equally distributed among subjectivity and context-domain classification respectively, that is each auxiliary task was sampled $\frac{1}{6}$ per epoch. Since additional recurrent encoding layers did not yield a significant rise in performance in MTL with AUX$_1$ (see Table 6.2), no experiments were performed where models leveraged a shared BiLSTM encoder.

What becomes apparent from the results depicted in Table 6.3, is the fact that OVERSAMPLING clearly outperformed UNIFORM SAMPLING in this MTL setting. I performed an independent *t*-test with respect to the results to test for statistical significance between the two task-sampling strategies. On average, models yielded significantly better exact-match accuracies and $F1$ scores in an OVERSAMPLING setting compared to a UNIFORM SAMPLING setting with $p < .05$. MTL concerning all three tasks, however, seems to be less helpful for the main task compared to sampling solely between QA and subjectivity classification. Note that none of the models could outperform the best model, BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple), and that on average performance was worse for this compared to the previous MTL set-up with just AUX$_1$.

Model \ Fine-tuning	SubjQA	
	Exact-match	$F1$
Auxiliary $\mathbf{1}$ & $\mathbf{2}$ - uniform sampling		
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$	75.24	75.85
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$ + adversarial (simple)	75.24	76.22
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$ + adversarial (GRL)	74.78	74.99
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$	75.45	75.75
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$ + adversarial (simple)	75.73	75.73
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$ + adversarial (GRL)	75.45	75.75
$\bar{\Theta}_{\mathbf{QA+Dom(q,c)+Sbj(q,\, a \vee c)}}$	75.32	75.72
Auxiliary $\mathbf{1}$ & $\mathbf{2}$ - oversampling		
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$	**76.16**	**76.49**
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$ + adversarial (simple)	76.10	76.46
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,c)}}$ + adversarial (GRL)	76.01	76.44
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$	76.01	76.44
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$ + adversarial (simple)	75.75	76.18
BERT$_{\mathbf{QA+Dom(q,c)+Sbj(q,a)}}$ + adversarial (GRL)	75.41	75.83
$\bar{\Theta}_{\mathbf{QA+Dom(q,c)+Sbj(q,\, a \vee c)}}$	75.91 *	76.31 *

Table 6.3: Multi-task learning (MTL) - Question Answering (QA) with two auxiliary tasks, namely subjectivity and context-domain classification. In a uniform sampling task setting, all tasks - main and auxiliary tasks - were randomly sampled according to a uniform distribution, whereas in an oversampling setting, the main task (i.e, QA) was sampled $\frac{2}{3}$ per epoch and $\frac{1}{3}$ was equally distributed among the aux tasks. Models were fine-tuned and evaluated on SubjQA or both SQuAD. Each model consisted of a shared pre-trained distilbert feature extractor and task-specific fully-connected output layers that were jointly fine-tuned on D_{subj}. Adversarial simple refers to adversarial training were the sign of the loss was simple reversed to make the model not learn the auxiliary task at all. Adversarial GRL refers to a more sophisticated adversarial strategy, namely a Gradient Reversal Layer (GRL) between the shared encoding layers and the task-specific output layers. * indicates a statistically significant difference between oversampling and uniform sampling according to an independent *t*-test a $\alpha < .05$.

6.1.5 Parallel Transfer: QA and Context-domain classification

In this MTL setting, mini-batches were alternated between QA (main) and context-domain classification (auxiliary). Hence, in a uniform sampling setting a model $\tilde{\mathbf{f}}$ was fine-tuned on QA in 50% of all training steps and on context-domain classification in the other half of training iterations. In an oversampling setting, QA was sampled $\frac{2}{3}$ per epoch and context-domain classification $\frac{1}{3}$ to make sure the model is exposed to the main task more frequently.

Interestingly, when compared against an MTL setting where mini-batches were alternated between QA and subjectivity classification, there was no statistical difference between the two MTL versions in a uniform sampling setting with respect to both the models' exact-match accuracies and $F1$-scores. However, with oversampling of the main task, MTL with aux$_1$ performed significantly better than MTL with aux$_2$ according to an independent *t*-test at $\alpha = .05$ where $p < .05$. Moreover, the current MTL set-up was the only MTL version where there was no difference between uniformly sampling tasks and oversampling the main task with respect to model performance (see Tables 6.2, 6.3, 6.4).

It seems as if in a uniform sampling setting, none of the two auxiliary tasks helped the model to en-

MODEL \ FINE-TUNING	SUBJQA	
	Exact-match	$F1$
Auxiliary $\mathbf{2}$ - UNIFORM SAMPLING		
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$	75.85	76.17
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$ + adversarial (simple)	**76.34**	76.48
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$ + adversarial (GRL)	75.85	76.17
$\bar{\Theta}_{\mathbf{QA+Dom(q,c)}}$	76.01	76.27
Auxiliary $\mathbf{2}$ - OVERSAMPLING		
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$	75.65	76.35
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$ + adversarial (simple)	75.77	**76.49**
$\mathrm{BERT}_{\mathbf{QA+Dom(q,c)}}$ + adversarial (GRL)	75.65	76.35
$\bar{\Theta}_{\mathbf{QA+Dom(q,c)}}$	75.69	76.40

Table 6.4: Multi-task learning (MTL) - Question Answering (QA) with context-domain classification as the only auxiliary task. In a UNIFORM SAMPLING task setting, all tasks - main and auxiliary task - were randomly sampled according to a uniform distribution, whereas in an OVERSAMPLING setting, the main task (i.e, QA) was sampled $\frac{2}{3}$ per epoch and $\frac{1}{3}$ was equally distributed among the AUX tasks. Models were fine-tuned and evaluated on SUBJQA. Each model consisted of a shared pre-trained DISTILBERT feature extractor and task-specific fully-connected output layers that were jointly fine-tuned on D_{subj}. ADVERSARIAL SIMPLE refers to adversarial training were the sign of the loss was simple reversed to make the model not learn the auxiliary task at all. ADVERSARIAL GRL refers to a more sophisticated adversarial strategy, namely a Gradient Reversal Layer (GRL) between the shared encoding layers and the task-specific output layers.

hance its performance on the main task. In contrast, in an OVERSAMPLING setting, subjectivity classification contributed to better QA performance on SubjQA, whereas context-domain classification did not help to answer subjective questions more accurately than STL. Models performed significantly worse in the latter MTL setting when compared to the former.

6.2 Sequential Transfer

Furthermore, I've trained and evaluated the baseline model, $\text{BERT}_{\mathbf{QA}}$, fine-tuned on all tasks in the task set $\mathbf{T} = \{T, T', T''\}$ sequentially until model convergence. Here, opposed to the MTL with parallel transfer setting, where HARD PARAMETER sharing is employed, knowledge transfer follows the rules of SOFT PARAMETER sharing (see Section 4.5).

As depicted in Table 6.5, training a model sequentially on all tasks did not enhance performance on the main task for most of the deployed training set-ups. It even deteriorated performance significantly, when the model received information trough an oracle, that is the concatenation of `hard` targets with the model's hidden representations for each input token at the last transformer layer. In the version where subjectivity classification was performed with respect to $(\mathbf{q}, \mathbf{c})$ input sequences performance did also decrease, but this time slightly rather than catastrophically. The only setting that yielded an increase in exact-match accuracy was the set-up, where `soft` targets, $\mathbf{p}_i \in \mathbf{R}^{k^s k^d}$, were concatenated with the hidden representations for each input token at the last transformer layer, $\mathbf{H}_i^6 \in \mathbf{R}^{T \times 768}$, and subjectivity classification was performed with respect to $(\mathbf{q}, \mathbf{a})$ input sequences (see Section 4.5 for methodological details w.r.t. the concatenation). None of the implemented sequential transfer models could contribute to an increase in $F1$.

One hypothesis to consider, is that the concatenation of `soft` targets - which encode probabilistic information about the auxiliary tasks - with the contextual hidden representations at the last transformer layer injected useful information about the natural language utterances in question and context into the model. Results have shown, however, that this additional information increased performance rather marginally.

MODEL \ FINE-TUNING	SUBJQA	
	Exact-match	$F1$
Auxiliary $\mathbf{1}$ & $\mathbf{2}$		
$\text{BERT}_{\mathbf{QA}}$ (`hard`)	63.25	65.58
$\text{BERT}_{\mathbf{QA}}$ (`soft`)	74.53	74.67
$\text{BERT}_{\mathbf{QA+Sbj(q,a)}}$ (`soft`)	**76.40**	76.40

Table 6.5: Sequential transfer across tasks. $\text{BERT}_{\mathbf{QA}}$ was fine-tuned sequentially on context-domain classification, subjectivity classification and QA respectively (in this order). For the main task, QA, the model received information either through an oracle, namely `hard` targets, or the other already converged learners, namely `soft` targets, about the previous tasks.

6.3 Fine-grained QA Results

The following investigations aimed at answering **RQ** 4 & 5. **RQ** 4 motivated the analysis of the difference in QA performance between review domains, whereas **RQ** 5 sought to infer the difficulty of subjective questions from interrogative words.

Domains

One crucial way to decipher a model's understanding of questions and their corresponding contexts is to examine its domain-specific performance if the dataset contains sentence pairs regarding various domains. What is interesting, is the observation that all evaluated models show a similar pattern concerning their domain-specific performance (see Table 6.6). Questions with respect to the domains `movies` and `books` were by far the easiest across the board. All evaluated models correctly predicted the answer span for $> 80\%$ and $> 79\%$ of questions regarding `movies` and `books` respectively. The difference in exact-match accuracies compared to the other four domains is more notable for a model that was fine-tuned on SQuAD. Recall that SQuAD consists of questions and paragraphs coming from a single domain only, namely `wikipedia` (see Table 5.3). It is fair to assume that Wikipedia contains more paragraphs about `movies` and `books` than it does about `restaurants`, `grocery` or `tripadvisor`. Hence, a model fine-tuned on SQuAD might have encountered more questions concerning `movies` and `books` than regarding the other domains which could partly explain the difference in results.

Questions concerning reviews about `tripadvisor` were clearly the most difficult for all evaluated models (see Table 6.6). This can in part be explained through the fact that despite reviews regarding `tripadvisor` appearing the least often in the train set, most test examples belonged to this domain - almost twice as many as from other domains (see Table 5.3).

The highest absolute and relative improvements of models fine-tuned on SubjQA over the baseline fine-tuned on SQuAD can be reported for the domains `grocery`, `restaurants` and `tripadvisor`. This might have a similar explanation as why the differences with respect to domain-specific performances are larger for a model fine-tuned on SQuAD (see above). The improvements of our best model, $\text{BERT}_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple), over the baseline, $\text{BERT}_{\mathbf{QA}}$, do not appear to be vast but are highest for the domains `grocery` and `tripadvisor`. Both domains can be considered to be among the set of the more difficult domains.

DOMAIN \ FINE-TUNING	SUBJQA		SQUAD
	BEST	BASELINE	BASELINE
`movies`	**87.59%** (+7.13%)	**87.59%** (+7.13%)	81.76%
`books`	**84.23%** (+6.34%)	84.00% (+6.05%)	79.21%
`electronics`	**81.46%** (+24.46%)	81.01% (+23.77%)	65.45%
`grocery`	**75.59%** (+44.89%)	74.58% (+42.96%)	52.17%
`restaurants`	**72.16%** (+68.13%)	71.81% (+67.31%)	42.92%
`tripadvisor`	**57.25%** (+98.65%)	55.88% (+93.89%)	28.82%
MEAN	**76.38%** (+30.81%)	75.81% (+29.83%)	58.39%

Table 6.6: Detailed exact-match accuracies for sentence pairs with respect to their review domain. Domains are sorted according to the respective model performance in descending order. Relative improvements over the baseline model fine-tuned on SQuAD (rightmost column) are depicted in parentheses. Best results are shown in bold face.

Interrogative words

To disentangle the source of the errors with respect to interrogative words, I've computed the exact-match accuracy for each question starting with one of the question words that are depicted in Table 5.5.

Table 6.7 shows that both models that were fine-tuned on SubjQA - BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple) and BERT$_{\mathbf{QA}}$ - achieved the highest relative improvements over BERT$_{\mathbf{QA}}$ fine-tuned on SQuAD for questions that started with `where` or `does`. This appears reasonable given the fact that both `where` and `does` are among the top-k interrogative words for the train set of SubjQA but not for SQuAD (see Table 5.5). Thus, a model that was fine-tuned on SubjQA has seen questions starting with `where` or `does` considerably more often than a model fine-tuned on SQuAD. What is surprising, however, is the fact that questions starting with `how` report the least performance gains, although such questions amount to $\approx 50\%$ of all questions in SubjQA, and represent only 8% of questions in SQuAD.

On the other hand, greater improvements can be reported for questions starting with `what` despite their significantly higher appearance in SQuAD compared to SubjQA. This might hint towards the post-hoc hypothesis of questions starting with `how` being "more" subjective than questions starting with `what` (according to Table 5.5), and thus are generally more difficult to answer than questions that start with `what`, as can be inferred from the poor scores for `how` and high scores for `what` questions across the board.

The best model, BERT$_{\mathbf{QA+Sbj(q,a)}}$ + adversarial (simple), improved over the baseline, BERT$_{\mathbf{QA}}$, mainly due to its enhanced performance regarding questions that start with `where`. For questions that start with one of the other top-k interrogative words the performance was not significantly different between the two model versions (see Table 6.7).

QUESTION WORD \ FINE-TUNING	SUBJQA		SQUAD
	BEST	BASELINE	BASELINE
`how`	**72.52**% (+28.01%)	72.27% (+27.57%)	56.65%
`what`	**79.38**% (+23.70%)	78.58% (+22.46%)	64.17%
`is`	**77.57**% (+29.37%)	77.15% (+28.67%)	59.96%
`where`	**79.55**% (+52.19%)	75.0% (+43.49%)	52.27%
`does`	**79.45**% (+39.19%)	78.99% (+38.38%)	57.08%
`do`	84.34% (+21.00%)	**85.35**% (+22.45%)	69.70%
MEAN	**78.80**% (+31.40%)	77.89% (+29.88%)	59.97%

Table 6.7: Detailed exact-match accuracies for sentence pairs whose questions start with one of the top-k interrogative words across both objective and subjective questions in SUBJQA (as depicted in Table 5.5). Relative improvements over the baseline model fine-tuned on SQUAD (rightmost column) are depicted in parentheses. Best results are shown in bold face.

6.4 Subjectivity Classification

6.4.1 Binary

To both better understand whether the different models show the ability to distinguish between subjective opinions and objective, measurable facts, and examine how difficult this particular auxiliary task (i.e., AUX$_1$) is in general, a learner must be optimized exclusively to classify question-context $(\mathbf{q}, \mathbf{c})$ or question-answer $(\mathbf{q}, \mathbf{a})$ pair sequences into subjective vs. objective. Therefore, BERT was additionally fine-tuned solely on sequence classification as the main task T without any other interfering task. Similarly to STL for QA, I fine-tuned BERT either on the train set of D_{Subj} or D_{Comb}, and evaluated the models on the test set of D_{Subj} only.

As in every other setting, each model consisted of a DISTILBERT feature-extractor, optional custom encoding layers, and two task-specific fully-connected linear output layers for binary sequence classification, one for

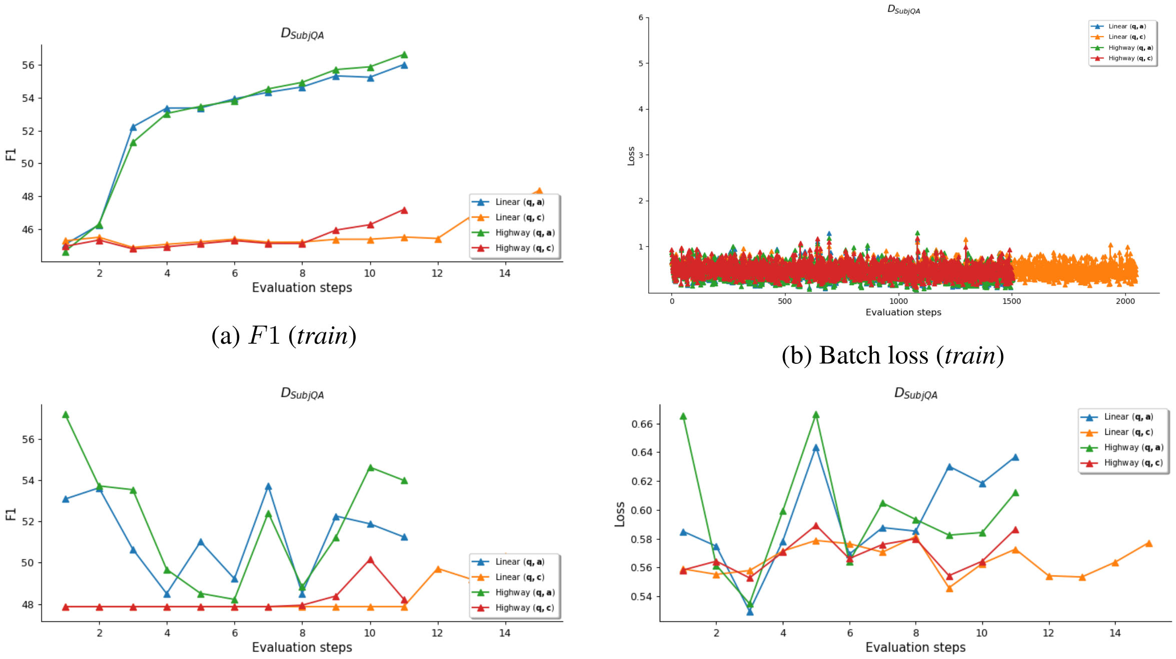
$\mathbf{q}$ and another for $\mathbf{c}$ or $\mathbf{a}$ depending on the set-up. Hence, the model had to classify both the question ($\mathbf{q}$) and its corresponding context ($\mathbf{c}$) or answer ($\mathbf{a}$) respectively into either a subjective opinion or an objective, measurable fact. Results are depicted in Table 6.8. Exclusively MACRO $F1$ scores are reported due to label imbalance (see Table 5.2). The accuracy score is not an appropriate metric to measure a classifier's performance on an imbalanced dataset since if a learner was to predict the majority class for every sample (e.g., exclusively subjective question-answer pairs), it would yield a high score without having the metric reflect whether the model did understand anything about the data whatsoever. I have leveraged MACRO instead of MICRO averaging for $F1$ as the latter takes class imbalance into account and thus does not deviate much from the accuracy score. In contrast, MACRO averaging reveals insights about a learner's performance with respect to all classes independent of label (im-)balance.

As can be inferred from Table 6.8, this task appeared to be difficult across the board. Not a single model yielded an $F1$-score $> 54.2\%$, which is not a particularly good results with respect to the general task of binary classification. The overall best model on this task, BERT$_{\mathbf{Sbj(q,a)}}$, achieved an $F1$-score of 54.17%, when fine-tuned on D_{subj} and 52.80% when fine-tuned on D_{comb} respectively. There is, however, no statistically significant difference with respect to the $F1$-scores between BERT$_{\mathbf{Sbj(q,a)}}$ and BERT$_{\mathbf{Sbj(q,a)}}$+ HIGHWAY according to an independent *t*-test at $\alpha = .05$. Whether the model was optimized to classify $(\mathbf{q}, \mathbf{c})$ or $(\mathbf{q}, \mathbf{a})$, however, made a notable impact on model performance. $\bar{\Theta}_{\mathbf{Sbj(q,a)}}$ performed better than $\bar{\Theta}_{\mathbf{Sbj(q,c)}}$ in both fine-tuning set-ups. The difference between the average models was even statistically significant when trained on D_{comb} at $\alpha = .05$ (see Table 6.8). This is in line with training and evaluation curves displayed in Figure 6.2, where both $F1$ scores on the train and development set respectively are higher and the minimum evaluation loss lower for models that were to classify $(\mathbf{q}, \mathbf{a})$ sequence pairs. For models that were optimized to classify $(\mathbf{q}, \mathbf{c})$ sequences it seems as if the models were not learning anything at all during training (see Figure 6.2).

Interestingly, an additional recurrent neural model, namely BiLSTM, on top of BERT that takes into account temporal dependencies between timesteps in a sequence of tokens $\mathbf{x}$, did not help on this task. It neither deteriorated nor enhanced the model's performance. Therefore, I did not report BERT + BiLSTM results in Figures and Tables respectively.

MODEL \ FINE-TUNING	SUBJQA	COMBINED
	$F1$	$F1$
BERT$_{\mathbf{Sbj(q,c)}}$	**54.17**	51.92
BERT$_{\mathbf{Sbj(q,c)}}$ + Highway	51.92	51.92
$\bar{\Theta}_{\mathbf{Sbj(q,c)}}$	53.05	51.92
BERT$_{\mathbf{Sbj(q,a)}}$	**54.17**	52.80
BERT$_{\mathbf{Sbj(q,a)}}$ + Highway	52.79	**53.07**
$\bar{\Theta}_{\mathbf{Sbj(q,a)}}$	53.47	52.94 *

Table 6.8: Subjectivity classification. Exclusively macro $F1$ scores reported due to class imbalance. Models were either fine-tuned on SUBJQA or both SQUAD and SUBJQA which I refer to as COMBINED, and evaluated on SUBJQA only. Each model consisted of a pre-trained DISTILBERT feature extractor and task-specific output layers for sequence classification that were fine-tuned on either of the two $D_{i \in \{subj,\, comb\}}$ versions. The abbreviation $(\mathbf{q}, \mathbf{c})$ refers to input sequences that consisted of question - context (i.e., question - review) sequence pairs, whereas $(\mathbf{q}, \mathbf{a})$ denotes question - answer pair input sequences.* indicates a statistically significant difference according to an independent *t*-test with $p < .05$.



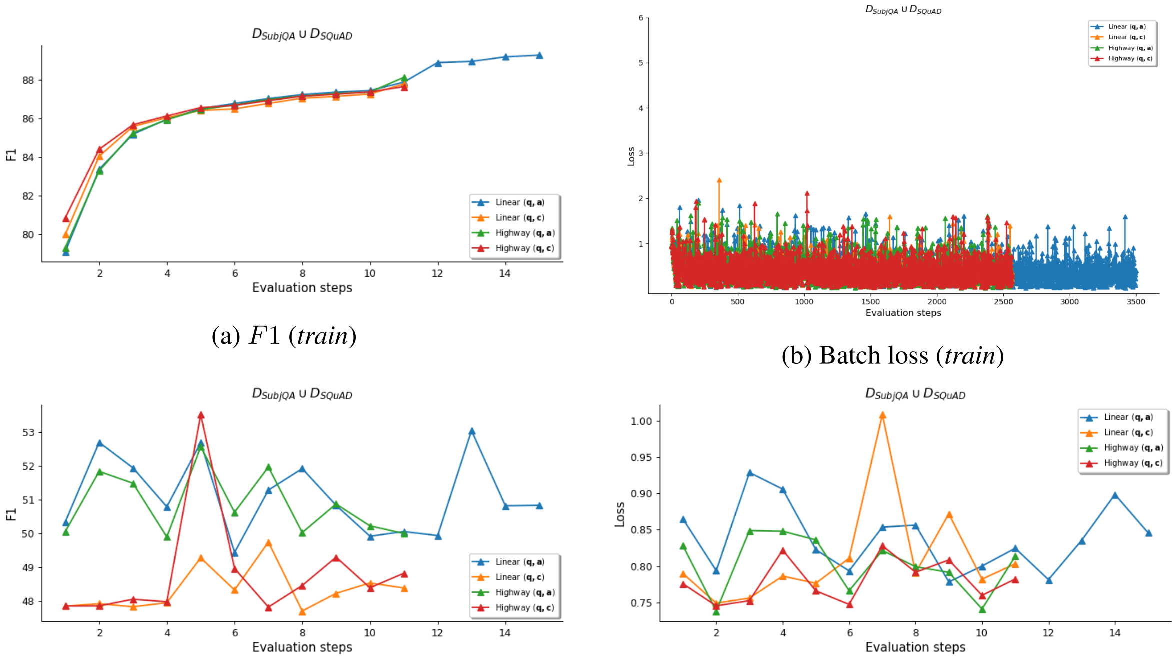
(a) $F1$ (*train*)

(b) Batch loss (*train*)

(c) $F1$ (*dev*)

(d) Evaluation loss (*dev*)

Figure 6.2: Subjectivity classification. Models were fine-tuned and evaluated on SubjQA. Depicted are $F1$ scores and cross-entropy losses as a function of evaluation steps for both train and development sets of D_{Subj} across all implemented STL models.



(a) $F1$ (*train*)

(b) Batch loss (*train*)

(c) $F1$ (*dev*)

(d) Evaluation loss (*dev*)

Figure 6.3: Subjectivity classification. Models were fine-tuned on both SQuAD and SubjQA which we call combined, and evaluated on SubjQA only. Depicted are $F1$ scores and cross-entropy losses as a function of evaluation steps for train and development sets of D_{Comb} and D_{Subj} respectively across all STL models.

The results indicate that it is highly difficult for a model to distinguish between word sequences that contain subjective linguistic signals $\mathbf{x_{subj}}$ and word sequences that include objective linguistic cues $\mathbf{x_{obj}}$. This is most probably since question-context pair sequences $(\mathbf{q}, \mathbf{c})$ on average consist of $100 - 300$ tokens per word sequence (see Figure 5.1) of which most are independent of subjective opinions or objective, measurable facts, and might therefore rather be domain- than opinion- or fact-specific. This is further reflected in the models' better classification performance on question-answer pair sequences $(\mathbf{q}, \mathbf{a})$ compared to question-context pair sequences $(\mathbf{q}, \mathbf{c})$ (see Table 6.8). It appears, however, that an answer's subjectivity might be reflected as part of the context and not in the context-free answer alone, which could explain the rather marginal improvements of classifying $(\mathbf{q}, \mathbf{a})$ over classifying $(\mathbf{q}, \mathbf{c})$, at least when both fine-tuned and evaluated on D_{subj}. That is another reason why I fine-tuned each model exclusively on context-domain classification as results with respect to subjectivity classification indicated that stronger signals might be reflected in the performance on the former task.

The following investigation was performed to conduct a more thorough error analysis. In so doing, I have examined whether the latter explanation is simply dataset specific, that is due to the linguistic nature of domain-variant reviews in D_{subj}. Hence, I fine-tuned the best model according to Table 6.8 exclusively on the train set of D_{comb} and evaluated the model on a synthetic test set $\in D_{comb}$ that consisted of the entire test set of D_{subj} and 10% of SQuAD's train set (see Table 5.2). The results of this analysis are displayed in Table 6.9.

MODEL \ FINE-TUNING	COMBINED		
	$(\mathbf{q}, \mathbf{c})^{\mathbf{sbj}}_{\mathbf{SubjQA}}$	$(\mathbf{q}, \mathbf{c})^{\mathbf{obj}}_{\mathbf{SubjQA}}$	$(\mathbf{q}, \mathbf{c})_{\mathbf{SQuAD}}$
BERT$_{\mathbf{Sbj(q,c)}}$	99.90%	0.00%	99.97%

Table 6.9: Fine-grained analysis of binary subjectivity classification. Depicted are accuracy scores per individual class. For this analysis, the objectivity class was split into question - context sentence pairs, $(\mathbf{q}, \mathbf{c})$, belonging to SUBJQA or SQUAD respectively.

The results in Table 6.9 show that the model did not understand the objective class $\in D_{Sbj}$ at all, and hence never made a correct classification with respect to sequences that belong to this class. The model most likely classified questions and answers respectively that belong to objective, measurable facts $\in D_{Sbj}$ into subjective opinions $\in D_{Sbj}$ as both sequence categories are part of the same dataset. To inspect whether the latter was the case or the model alternatively believes objective $(\mathbf{q}, \mathbf{a} \vee \mathbf{c})$ sequences $\in D_{Sbj}$ belong to D_{Obj} which is entirely objective, one must conduct a multi-way sequence classification experiment, and in so doing perform a thorough analysis of both the model's predictions and hidden representations in latent space with respect to each class.

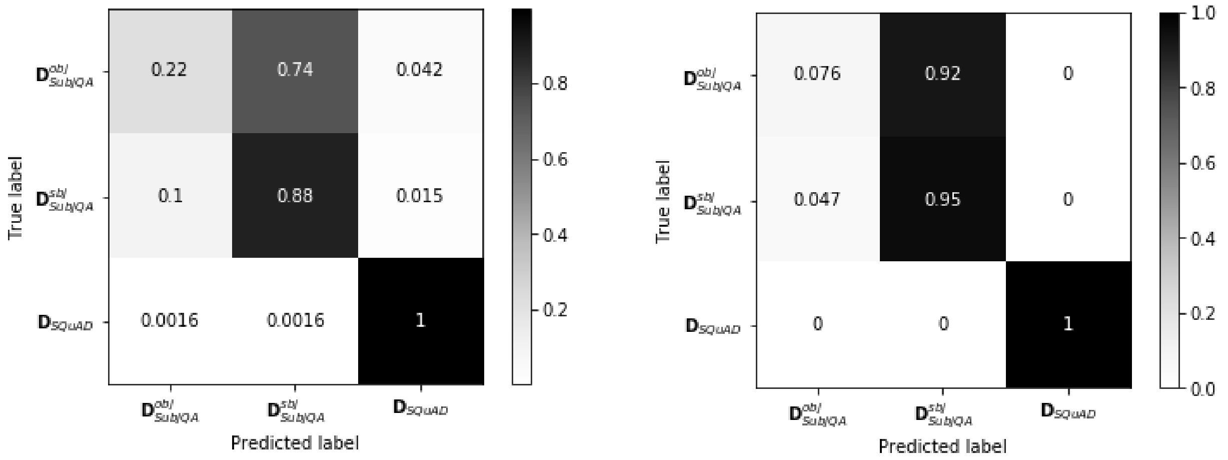
The latter step is crucial to equip the model with the possibility to learn individual representations for each of the three categories, which is not possible in the binary classification task, where objective questions are encoded with the same label no matter whether they belong to SUBJQA or SQUAD, and therefore optimized to learn feature representations for two different classes. Moreover, $F1$ scores concerning the binary classification task, where evaluation was performed on the synthetically created dataset, are not reliable since the model generally performed well on the objective class. Recall that all objective questions belong to the same class. Thus, the high $F1$ score of 87.24% is not an adequate reflection of the model's comprehension of the three classes.

6.4.2 Multi-way

To decipher the complexity of distinguishing between subjective and objective questions in SubjQA, I transformed the aforementioned subjectivity classification task from a binary into a multi-way classification problem. This time, the model was required to not only classify whether a question $\mathbf{q}_i$ was subjective or objective but had to differentiate between both subjective questions $\in$ SubjQA, objective questions $\in$ SubjQA, and objective questions $\in$ SQuAD. Hence, the model was trained to learn three classes instead of two, that is $(\mathbf{q}, \mathbf{a} \vee \mathbf{c}) \in \{D^{obj}_{Sbj}, D^{sbj}_{Sbj}, D_{Obj}\}$.

This experiment was conducted to investigate whether the difficulty of distinguishing between subjective and objective questions is dataset-specific, that is owing to the linguistic nature of SubjQA, or a general one, that is due to the model's inability of understanding the semantic differences between subjective and objective questions. As mentioned in the previous section, SQuAD is a dataset that consists entirely of objective questions and answers extracted from Wikipedia paragraphs [62, 61]. Hence, if the model is indeed not capable of differentiating between subjective and objective questions, it will not perform well on the synthetically added SQuAD class either. Else, the poor performance concerning the binary classification task lies in the dataset and not in the capacity of the model. As in every training set-up, weighting of the loss with respect to each class was applied accordingly to account for label imbalance.

As can be inferred from the task-specific confusion matrices (see Figure 6.4), the model was lacking the ability to distinguish between subjective and objective questions in SubjQA. The poor performances with respect to objective questions $\in$ SubjQA are in line with the binary classification results depicted in Table 6.9. The model could, however, perfectly differentiate between questions from SubjQA and SQuAD respectively. The model almost always predicted the subjective class for objective questions from SubjQA when classifying question - context, $(\mathbf{q}, \mathbf{c})$, pair sequences (see Figure 6.4b). It did a slightly better job in differentiating subjective from objective questions in SubjQA when classifying question - answer, $(\mathbf{q}, \mathbf{a})$, pair sequences (see Figure 6.4a). The latter task yielded with 68.26 macro $F1$ a 3.57% relative improvement over 65.91 macro $F1$ in the former task, which is a considerable enhancement but still not an incredibly high performance. The difference between the two classification set-ups becomes more apparent in the 2D t-SNE plots of each sentence pair's semantic representation. The latter is visually depicted in Section 7.



(a) Multi-way $(\mathbf{q}, \mathbf{a})$ classification

(b) Multi-way $(\mathbf{q}, \mathbf{c})$ classification

Figure 6.4: Normalized confusion matrices. The principal diagonal of the matrix depicts class-specific recall scores. The higher the score in the diagonal, the better did the model perform with respect to the corresponding class.

6.5 Context-domain Classification

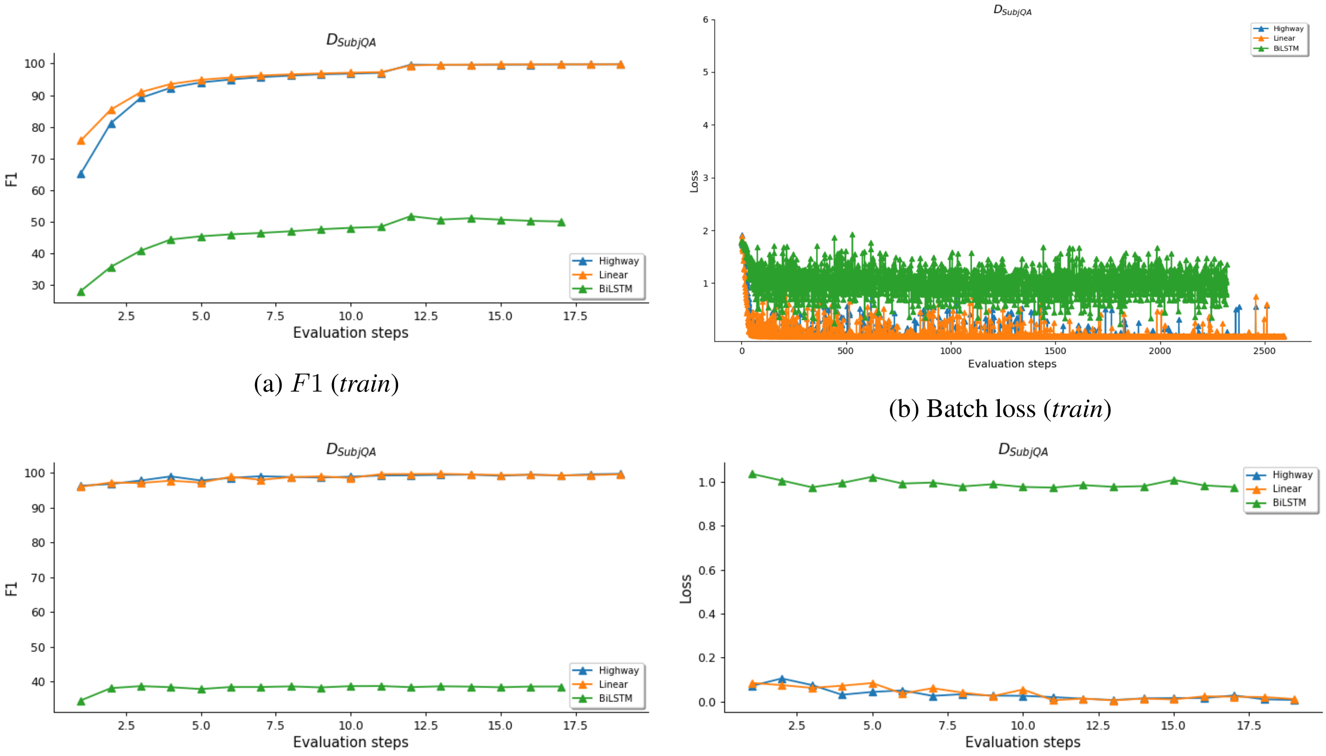
To inspect whether the second auxiliary task AUX$_2$, namely context-domain classification, is useful to enhance QA with respect to reviews that belong to different domains, each of the STL models was exclusively fine-tuned on context-domain classification. As is depicted in Table 6.10, it was fairly easy for the STL models, BERT and BERT + HIGHWAY in particular, to classify the questions and their corresponding reviews $(\mathbf{q}, \mathbf{c})$ into their respective domains $(\mathbf{y}^d)$.

In contrast to subjectivity classification, this time both accuracy and (macro) $F1$-scores are reported since both train and development sets were fairly balanced for domain labels (see Table 5.3). If a model $\tilde{\mathbf{f}}$ was trained on D_{comb}, domains were weighted accordingly (see Section 4.2 for further details). BERT and BERT + HIGHWAY achieved a macro $F1$ and an accuracy score of $> 98\%$ and $> 99\%$ respectively in any fine-tuning setting (see Table 6.10) - no matter whether the model was fine-tuned on D_{subj} or D_{comb}. This means, that context-domains $(\mathbf{y}^d)$ contain insightful linguistic signals that could lead to different results when answering questions about reviews from different domains, and hence might serve as a relevant auxiliary task in an MTL setting with respect to QA

What's interesting, however, is the fact that, in contrast to QA, where an additional RNN model between BERT and the fully-connected QA output layer helped (see Table 6.1), a BiLSTM between BERT and the linear classification layer deteriorated the model's performance by an order of magnitude. I suspect this is because the special [CLS] token in any BERT model encodes semantic information of the entire sentence $(\mathbf{x})$ or sentence pair sequence $(\mathbf{q}, \mathbf{c})$ respectively [21], but is not used to classify a sequence $(\mathbf{x})$ in an RNN based model. Any model based on recurrencies - opposed to linear connections that do not take the order of each token x_i in a sequence $\mathbf{x}$ into account - encodes a sequence $\mathbf{x}$ timestep by timestep while taking into account each of the previous timesteps $(t-1, t-2, ..., t-i)$. Therefore, in an RNN based model the vector at the last timestep $\mathbf{t}$ (i.e., last index in a matrix of continuous-valued vectors), is used as the input to the linear output layer to perform the classification task. This might result in a vector that does not encode the semantic information of the entire sequence thoroughly when used on top of BERT, since the special [CLS] token alone does already exploit this linguistic information. It is yet interesting to inspect why the latter computation is not feasible and decipher which timestep in an RNN module encodes the information contained in [CLS]. I leave this investigation for future work and encourage others to look deeper into this conundrum.

MODEL \ FINE-TUNING	SUBJQA		COMBINED	
	Accuracy	$F1$	Accuracy	$F1$
BERT$_{\mathbf{Dom(q,c)}}$	99.23	**98.65**	**99.49**	**98.89**
BERT$_{\mathbf{Dom(q,c)}}$ + Highway	**99.25**	98.61	99.37	98.75
BERT$_{\mathbf{Dom(q,c)}}$ + BiLSTM	45.58	28.62	40.73	24.52

Table 6.10: Context-domain classification. Models were either fine-tuned on SUBJQA or both SQUAD and SUBJQA which I refert to as COMBINED, and evaluated on SUBJQA only. Each model consisted of a pre-trained DISTILBERT feature extractor, custom encoding layers on top of BERT and task-specific output layers for multi-class sequence classification that were fine-tuned on either of the two $D_{i \in \{subj,\, comb\}}$ versions. The abbreviation $(\mathbf{q}, \mathbf{c})$ refers to input sequences that consisted of question - context (i.e., question - review) sequence pairs.



(a) $F1$ (*train*)

(b) Batch loss (*train*)

(c) $F1$ (*dev*)

(d) Evaluation loss (*dev*)

Figure 6.5: Context-domain classification. Models were fine-tuned on SUBJQA, and evaluated on SUBJQA. Depicted are $F1$ scores and cross-entropy losses as a function of evaluation steps for both train and development sets of D_{Subj} across all implemented models.

Chapter 7

Qualitative Analyses

7.1 Hidden Representations in Latent Space

To decipher the performance, or more generally the behavior, of any neural network architecture, one is required to inspect a model's hidden representations at each of its spatial dimensions (opposed to a model's temporal dimensions such as in sequence modelling tasks with respect to temporal data). In neural networks, spatial dimensions or stages, are represented through layers. The more layers a neural network consists of, the deeper it is in space [43]. Each layer refers to a different stage within the model's representational hierarchy. Usually, in earlier stages of a neural network, that is in the bottom layers, the model's representations in latent space draw attention to low-level features such as syntax in NLP models (e.g., Part-of-Speech tags) [11] or edges in Computer Vision (CV) models [34, 41, 71, 24]. Contrary, later stages in the network, that is the top layers, focus on the representation of high-level features such as meaning in NLP models (e.g., relations between entities or the meaning of a word dependent on the context it appears in) [38, 46, 45, 21] or abstract visual features in CV models (e.g., eyes, nose, mouth in faces) [39, 41, 43].

According to this general idea of a representational hierarchy in space, the notions of `question` and `answer` may be reflected in the top layers rather than in the early layers of the neural model. In this section, I will look deeper into the hidden representations of selected model architectures and examine what qualitative insights may be gained from those, why some classifications did not work as expected and most importantly at which stages in the hierarchy the model made mistakes. This investigation is dedicated to answer **RQ** 6.

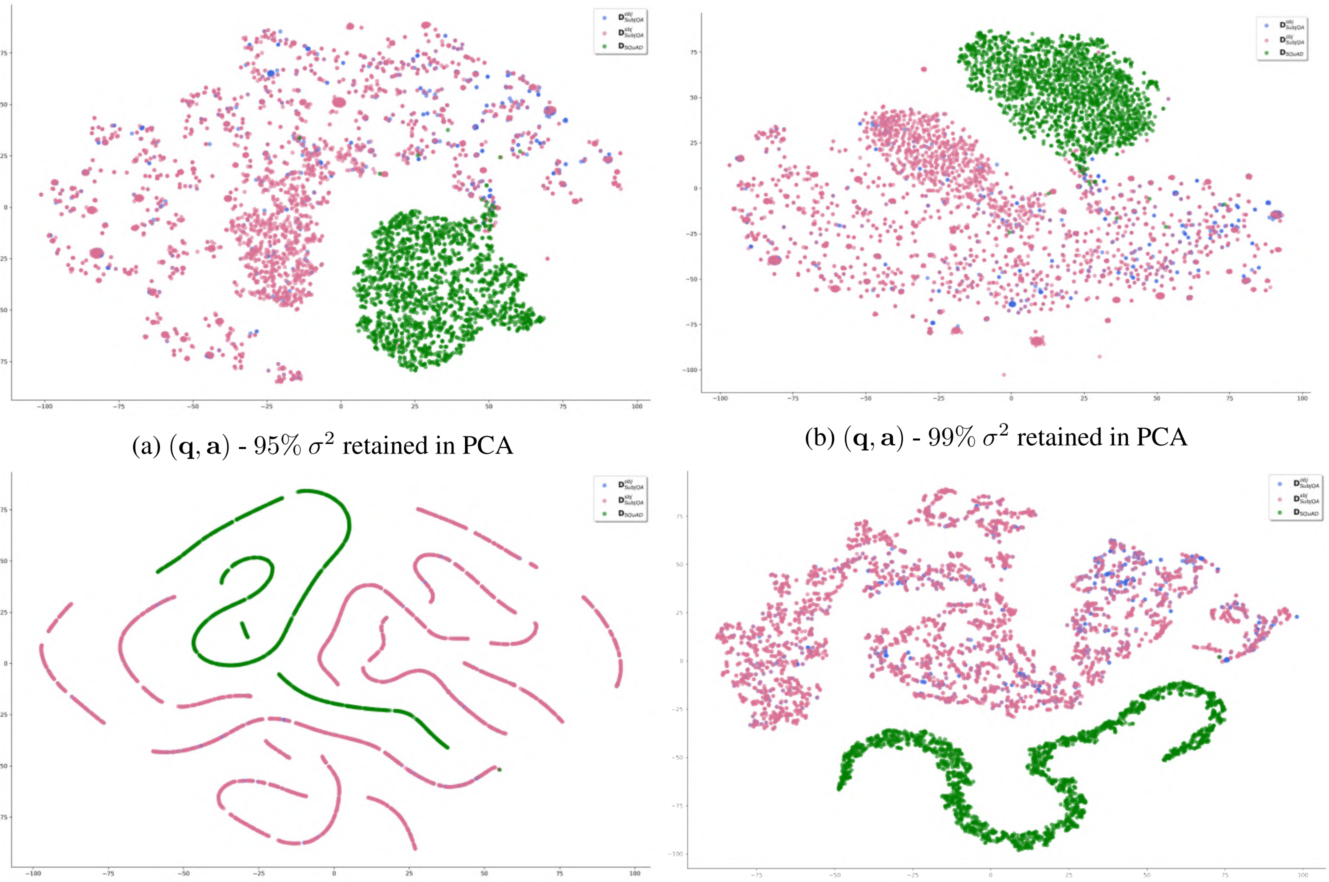
7.2 Multi-way Subjectivity Classification

Firstly, I will shed light on the task of multi-way **subjectivity classification** to better understand how subjectivity is reflected in the model's hidden representations. According to the **quantitative results** with respect to this task, the model seemingly could not distinguish between subjective and objective questions within $\mathbf{D}_{SubjQA}$, particularly when trained and evaluated on question - context pair sequences (see Figure 6.4).

To better understand why this is the case, I first projected each sentence pair's semantic representation in vector space - which is reflected in the hidden representation w.r.t. BERT's special `[CLS]` token - for each sentence pair with Principal Components Analysis (PCA) [72] onto n principal components that either retained 95% or 99% of the variance prior to transforming the sentence embeddings into 2D space [1]. The former step was performed to both save computational time - t-SNE is an expensive algorithm that leverages stochastic gradient descent (SGD) to iteratively search for a low-dimensional feature space until convergence [51] - and examine differences in the projections dependent on the retained variance. As clearly reflected in BERT's low dimensional sentence pair projections, questions from SQuAD are clustered in a space that is highly distinguishable from the two other classes, both of which belong to SubjQA (see Figure 7.1).

[1] Using the t-Distributed Stochastic Neighbor Embedding (t-SNE) implementation provided by scikit-learn [55]

This holds even more so when classifying $(\mathbf{q}, \mathbf{c})$ as indicated by the quantitative results (see Figure 6.4). The model's feature representations for $(\mathbf{q}, \mathbf{c})$ sentence pairs that belong to SQuAD are embedded in a space that is perfectly distinguishable from the two other classes. In those projections, the data points do not even touch one another as opposed to t-SNE plots with respect to $(\mathbf{q}, \mathbf{a})$ sentence pairs. However, the model does not appear to differentiate SubjQA's subjective from SubjQA's objective questions in latent space whatsoever. This is in line with the class-specific results depicted in the confusion matrices 6.4, and hints towards the potential post hoc hypotheses that either none of the questions and corresponding answers in SubjQA is fully objective or subjective and thus cannot be modeled with discrete values, or SubjQA is in general a relatively subjective dataset. I will elaborate this in more detail in Section 8.



(a) $(\mathbf{q}, \mathbf{a})$ - 95% σ^2 retained in PCA

(b) $(\mathbf{q}, \mathbf{a})$ - 99% σ^2 retained in PCA

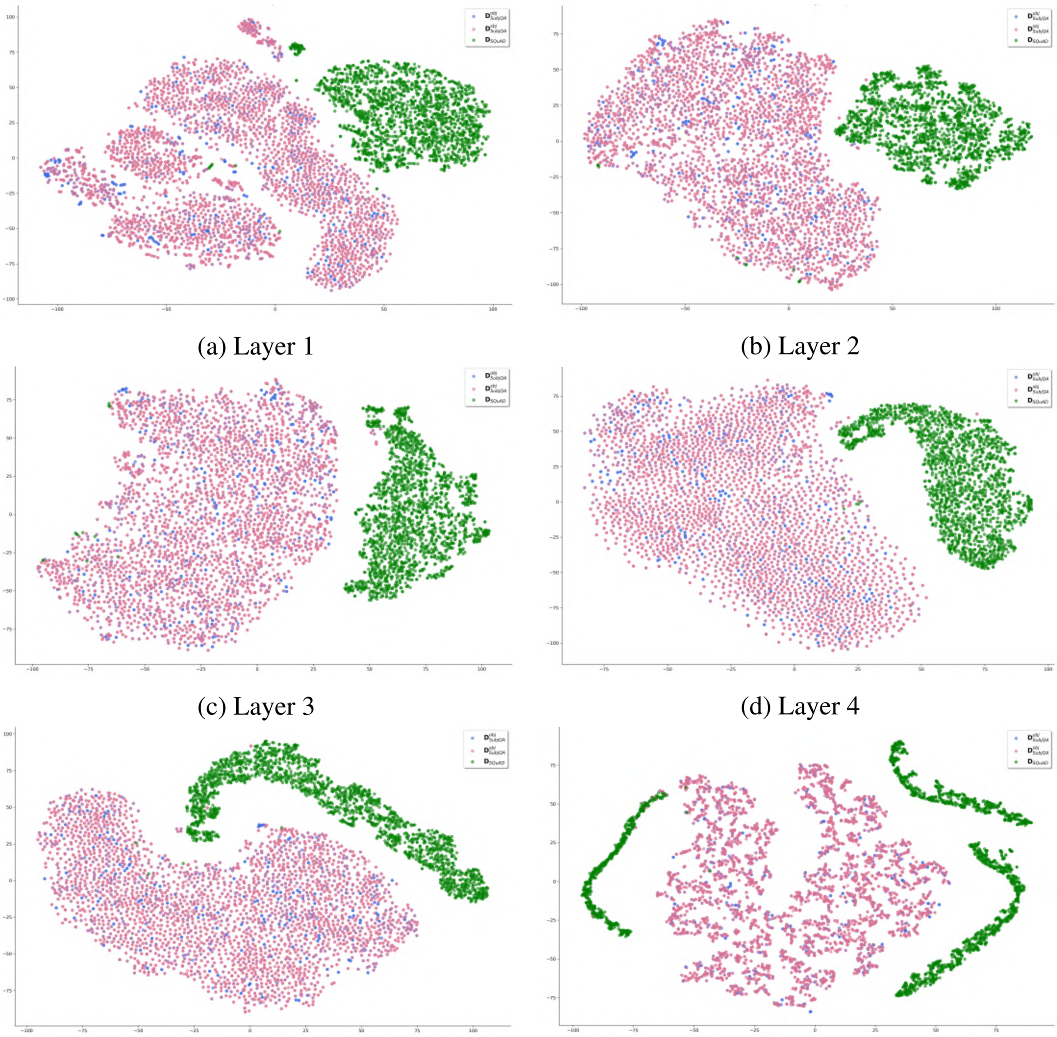
(c) $(\mathbf{q}, \mathbf{c})$ - 95% σ^2 retained in PCA

(d) $(\mathbf{q}, \mathbf{c})$ - 99% σ^2 retained in PCA

Figure 7.1: BERT's feature representations of $(\mathbf{q}, \mathbf{a})$ and $(\mathbf{q}, \mathbf{c})$ sequences in latent space - iteratively optimized during multi-way classification - projected onto 2D via PCA and t-SNE respectively. The upper and lower row display question - answer sentence pairs, that is $(\mathbf{q}, \mathbf{a})$ sequences, and question - context sentence pairs, that is $(\mathbf{q}, \mathbf{c})$ sequences, respectively. Each sentence pair was represented through BERT's semantic representation of the entire sequence reflected in the special `[CLS]` token. Feature representations were first transformed via PCA into d-dimensional space (depending on σ^2) and then projected onto 2D via t-SNE. Pink: subjective questions $\in \mathbf{D}_{SubjQA}$. Blue: objective questions $\in \mathbf{D}_{SubjQA}$. Green: objective questions $\in \mathbf{D}_{SQuAD}$. Depicted are hidden representations from the model's last (6th) layer.

7.3 Multi-task Learning for Question Answering

Knowing about subjectivity but being dataset agnostic One could argue that in the (multi-way) subjectivity classification task the model somehow learned to distinguish SubjQA from SQuAD. Thus, subjective and objective questions that belong to SUBJQA were projected into the same latent space but examples that belong to SQUAD were separated from the latter two. To account for this potential objection, I've implemented a model that was simultaneously trained on three tasks, namely QA, subjectivity, and dataset classification, of which the latter classification task was performed in an adversarial manner with a Gradient Reversal Layer (GRL) [27] in between BERT and the respective task-specific output layer(s) (see Section 4.4 for further details). Hence, the model was optimized to correctly classify questions into subjective vs. objective as was done in the previous setting - although not in a multi-way but binary fashion with 1 representing the subjective and 0 the objective class - but at the same time trained to not know anything about the source of the sentence pair example, that is being dataset agnostic.



(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Layer 4

(e) Layer 5

(f) Layer 6

Figure 7.2: Dataset agnostic MTL model fine-tuned on $\mathbf{D}_{SubjQA} \cup \mathbf{D}_{SQuAD}$. Depicted are the model's hidden representation w.r.t. BERT's special `[CLS]` token at each layer for every sentence pair example in the combined test set $\mathbf{D}_{comb}$. Feature representations across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1st) (top-left) and stopping at the last (6th) (bottom-right) layer. Retained variance in PCA: 99%. Pink: subjective questions $\in \mathbf{D}_{SubjQA}$. Blue: objective questions $\in \mathbf{D}_{SubjQA}$. Green: objective questions $\in \mathbf{D}_{SQuAD}$.

The MTL training was performed in an ALTERNATING batch setting, where examples for subjectivity classification were drawn from batches that consisted of $(\mathbf{q}, \mathbf{a})$ sequences, whereas the model received $(\mathbf{q}, \mathbf{c})$ sequences as inputs for all other tasks. This was done, to ensure that the model does not learn the subjectivity classification task dependent on contextual features but question and answer tokens alone. Moreover, **quantitative results** have shown that solely classifying $(\mathbf{q}, \mathbf{a})$ sequence pairs yields better results than classifying $(\mathbf{q}, \mathbf{c})$ sequences (see Table 6.8). Note that the subjectivity classification task was optimized as a binary and not as a multi-way classification task. The objective class was synthetically split into SQUAD and SUBJQA post hoc. This time, the model's hidden representations are depicted for each layer to investigate whether differences between objective and subjective questions occur exclusively at later or even at earlier stages of the neural network.

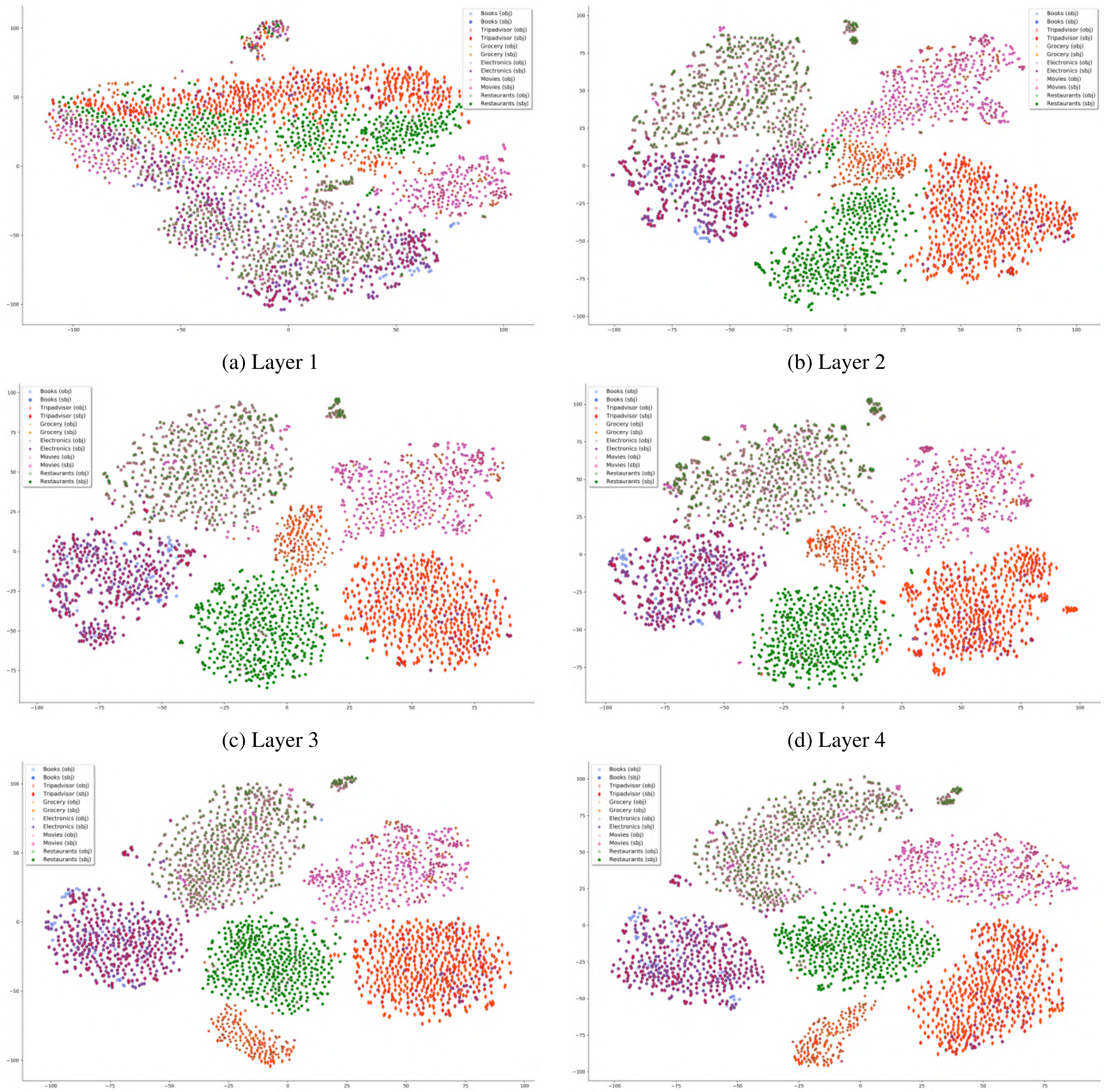
As clearly indicated by the model's hidden representations projected into $\mathbf{R}^2$ (see Figure 7.2), the differences in the linguistic signals between objective and subjective questions $\in \mathbf{D}_{SubjQA}$ appear to be too marginal to be distinguished from one another. Objective questions that belong to SQUAD, however, are embedded in a notably different part of the vector space. The latter becomes apparent even in the 1st layer of the network (see Figure 7.2 a). This indicates that the model is simply not able to differentiate between objective and subjective questions within SUBJQA, even if the model is trained adversarially to be agnostic concerning the source of the data point, but can easily separate objective questions that belong to SQUAD from any question that belongs to SUBJQA in latent space.

7.4 Sequential Transfer for Question Answering

Inspecting sequentially transferred representations To further investigate into the inability of the model to distinguish subjective from objective questions within SUBJQA, I examined the hidden representations of a sequential transfer model that was sequentially trained on all tasks, namely QA, context-domain and subjectivity classification, until convergence (see Section 4.5 for implementation details). To ensure that the model performed subjectivity classification prior to QA, the inspected model was firstly optimized on context-domain classification, followed by subjectivity classification and QA. Similarly to the MTL model, subjectivity classification was performed with batches that contained $(\mathbf{q}, \mathbf{a})$ sequences to ensure that the classification is done solely with respect to question and answer tokens respectively and not interfered by linguistic signals of the context.

As can be inferred from the model's feature representations projected into $\mathbf{R}^2$ (see Figure 7.3), the model could to a large extent separate the domains from each other - although overlaps between certain domains can be observed -, but could on the other hand not distinguish between subjective and objective questions within domains. Recall that context-domain classification was performed as the first task in the task sequence $\mathbf{T} = [T, T', T'']$, that is prior to subjectivity classification. Since context-domain clusters are visible in the 2D projections even after the model converged on all three tasks sequentially (see Figure 7.3), one may draw two (alternative) conclusions that are not mutually exclusive.

Firstly, the linguistic signals extracted from review domains are both strong enough to be retained after finetuning on two other tasks and more notable than signals extracted from the subjectivity classification part. Secondly, $(\mathbf{q}, \mathbf{c})$ sentence pairs are distinguishable due to their context-domains but not because of their difference in subjectivity levels labeled through human crowd workers (see Figure 5.2 for a depiction of the subjectivity level distributions across domains). The latter indicates what has been observed in the sections above, namely that the difference in linguistic signals between objective and subjective questions in SubjQA appears to be too weak to be modeled.



(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Layer 4

(e) Layer 5

(f) Layer 6

Figure 7.3: Sequential transfer model fine-tuned exclusively on $\mathbf{D}_{SubjQA}$. Depicted are the model's hidden representation w.r.t. BERT's special `[CLS]` token at each layer for every sentence pair example in SubjQA's test set $\mathbf{D}_{comb}$. Feature representations across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1st) (top-left) and stopping at the last (6th) (bottom-right) layer. Blue: `books`. Red: `tripadvisor`. Dark orange: `grocery`. Indigo: `electronics`. Pink: `movies`. Green: `restaurants`. Within each domain, colors with higher intensity represent **subjective** and lower intensity colors objective questions. Retained variance in PCA: 99%.

7.5 Error Analysis

This section is entirely dedicated to the understanding of general error sources concerning QA. To enhance a network's performance and improve its learning, it is crucial to examine `which` errors a model made, and `why` those errors happened in the first place.

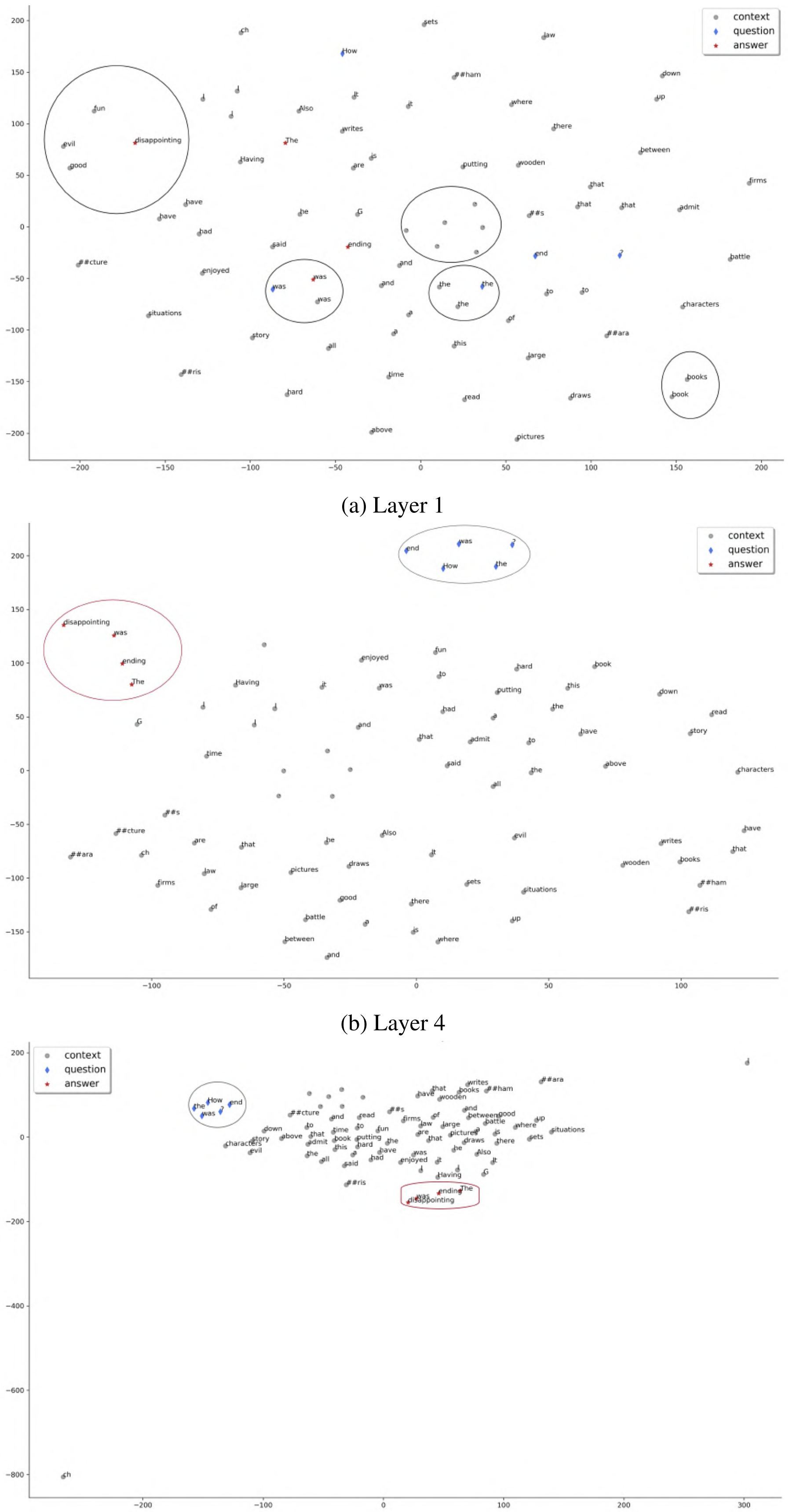
7.5.1 Question Answering in Vector Space

Solely inspecting erroneous predictions regarding the question type (e.g., objective vs. subjective) or the corresponding domain (e.g., movies vs. grocery), does not yield insights into `why` and `where` along the way a learner made mistakes. Therefore, I've deviated from the usual error analysis, and instead of just showing examples of correct and erroneous predictions in the form natural language text taken a slightly different approach, inspired by one recently published paper [2]. This study has for the first time analyzed BERT's hidden representations after performing QA and thus contributed to a more thorough understanding of the inner-workings of Transformers [64] (see Section 2.1 for more information). Following the approach of [2], I've investigated the model's hidden representations at each layer for every token in a randomly chosen sentence pair. In so doing, I've projected them - similarly to the visualizations of hidden states concerning the different classes - with PCA and t-SNE from $\mathbf{R}^{768}$ into $\mathbf{R}^2$. This layer-wise analysis reveals information about the model's clustering of natural language utterances in latent space at each stage of the model at inference time.

To yield visualizations of hidden states for each token in a word sequence, I've chosen one random sentence pair among the following three sets: correct predictions w.r.t. answerable questions, correct predictions w.r.t. unanswerable questions, erroneous predictions w.r.t. answerable questions. The former and the latter set reveal particular insights into `why` a model made a wrong prediction.

Figure 7.4 illustrates hidden states for every token in a randomly chosen sentence pair for which the model correctly answered the questions. Depicted are representations for layers 1, 4, and 5. One can see that low-level features concerning language are depicted in the first layer. Here, tokens that are generally syntactically or semantically similar are clustered together. For instance, definite determiners such as `the`, indefinite determiners such as `a` or conjunctions such as `and` are each clustered in a similar space that is distinguishable from the other linguistic classes. Hence, the model has not yet grasped the high-level concept of question and answer but draws attention to the general features of natural language. In layer 4 the model has projected all tokens that belong to either the question, the answer, or the context into a similar latent space. This indicates an understanding of high-level features with respect to the notion of question and answer. The same holds for layer 5. What is compelling, is that the model clustered both the question and the answer in a separate space from the context even in layer 4. This could indicate that knowledge from later layers might not be necessary to answer the question.

Looking at Figure 7.5 unveils that it was fairly easy to separate the vector representation of the special `[CLS]` token from the rest of the text. Recall that the `[CLS]` token must be predicted, if and only if a question is not answerable from the given context. The model is therefore required to only predict a single token which by the mere length of the answer span is easier than predicting multiple tokens. Again, tokens corresponding to the question were projected into a space that is distinguishable from the spatial representations of context and answer.

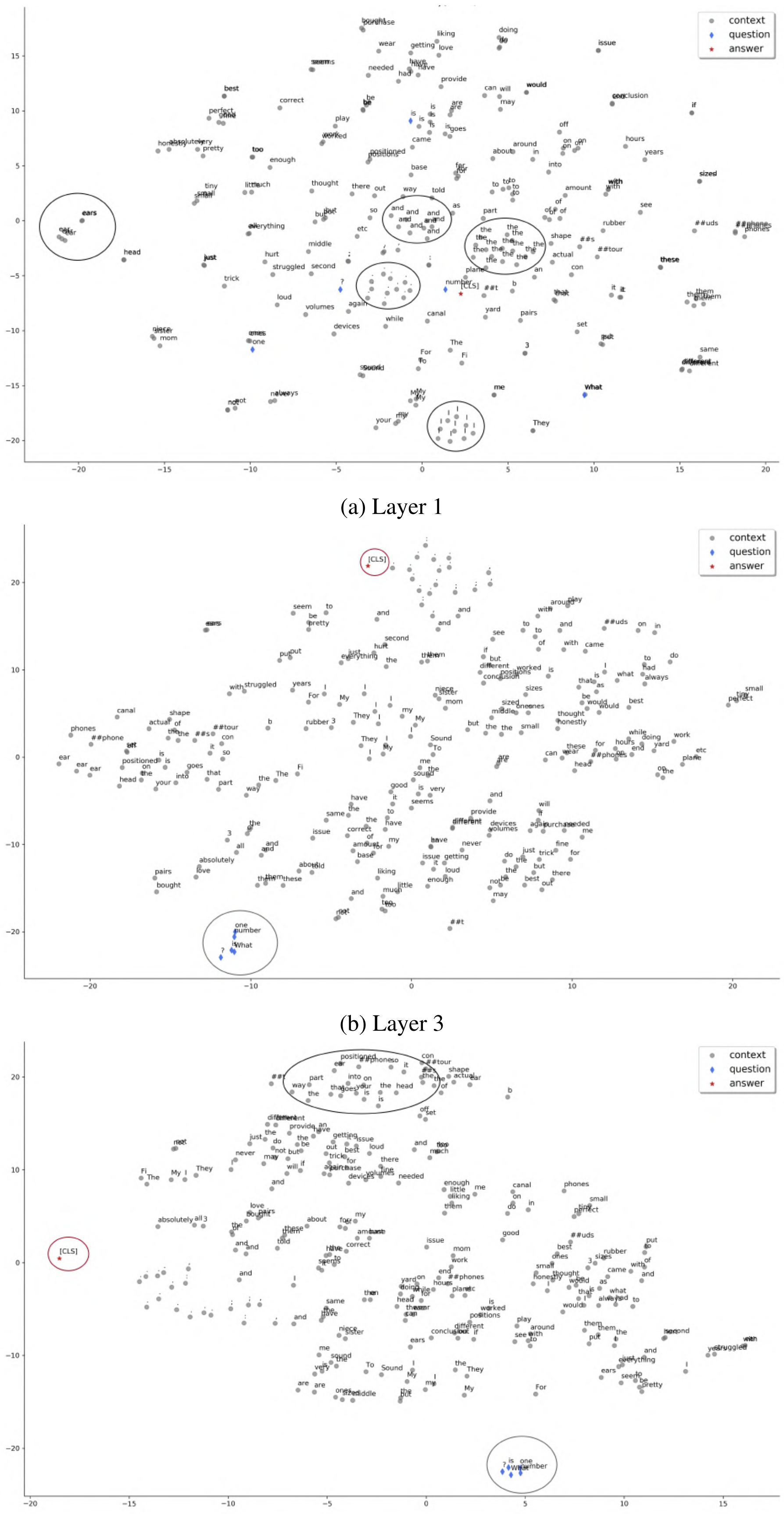


(a) Layer 1

(b) Layer 4

(c) Layer 5

Figure 7.4: Correct answer-span prediction for an answerable question. Depicted are BERT's hidden representations at different stages of the model from bottom to top (i.e., Layer 1, 4 - 5) projected into $\mathbf{R}^2$ for every token in a randomly chosen input sequence (q, c)$_i$ among the set of sentence pairs for which the model correctly predicted the answer-span a$_i$ w.r.t. q$_i \in \mathbf{q}_{answerable}$. Blue diamonds: question. Red stars: answer. Grey circles: context. (Q: "how was the end?", A: "The ending was disappointing")

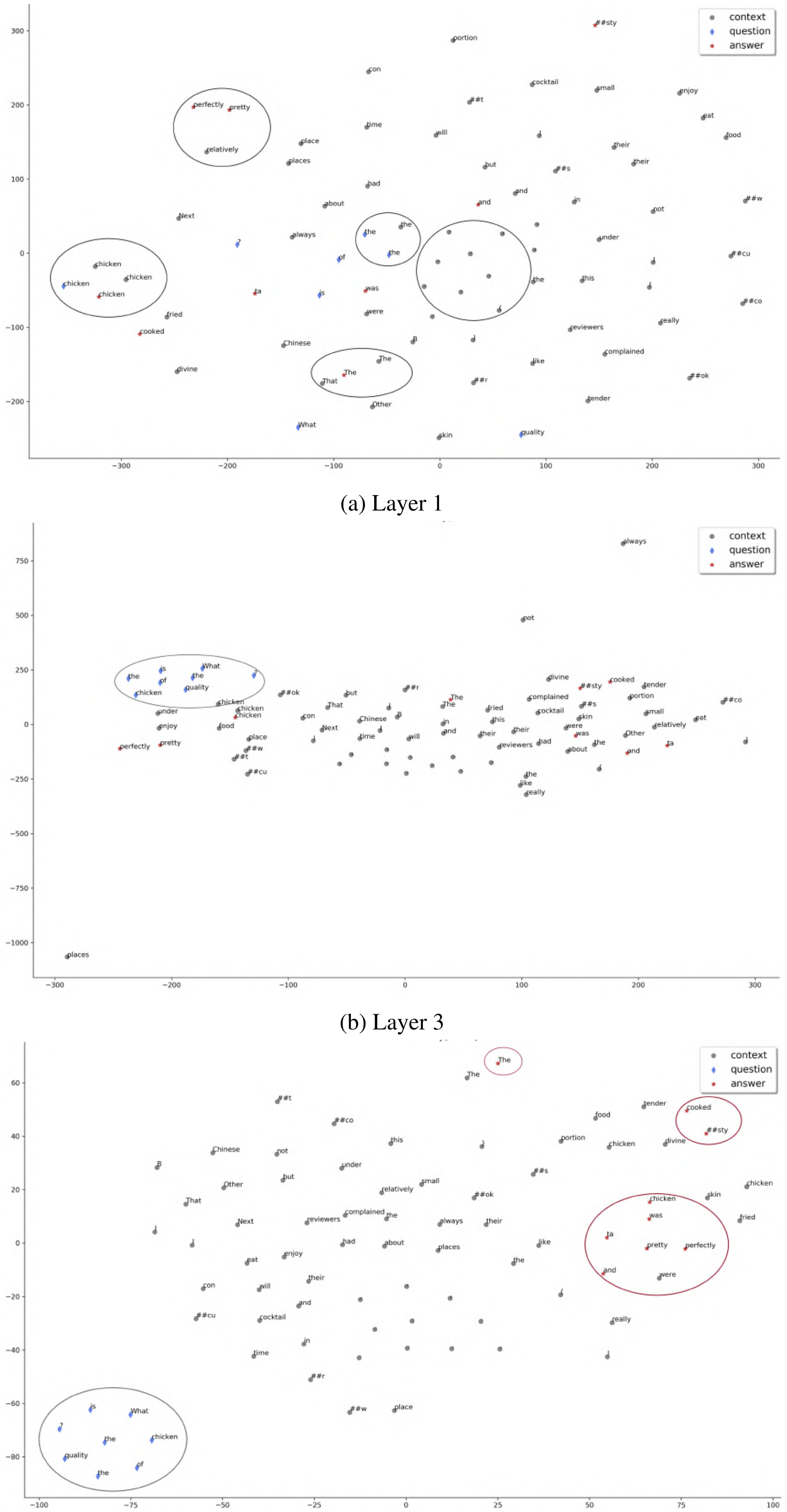


(a) Layer 1

(b) Layer 3

(c) Layer 5

Figure 7.5: Correct answer-span prediction for an unanswerable question. Depicted are BERT's hidden representations at different stages of the model from bottom to top (i.e., Layer 1, 3, 5) projected into $\mathbf{R}^2$ for every token in a randomly chosen input sequence (q, c)$_i$ among the set of sentence pairs for which the model correctly predicted the special `[CLS]` token w.r.t. $q_i \in \mathbf{q}_{unanswerable}$. Blue diamonds: question. Red stars: answer. Grey circles: context. (Q: "`what` is number one?", A: `[CLS]`)



(a) Layer 1

(b) Layer 3

(c) Layer 5

Figure 7.6: Erroneous answer-span prediction for an answerable question. Depicted are BERT's hidden representations at different stages of the model from bottom to top (i.e., Layer 1, 3, 5) projected into $\mathbf{R}^2$ for every token in a randomly chosen input sequence $(\mathrm{q}, \mathrm{c})_i$ among the set of sentence pairs for which the model could not predict the answer-span a_i w.r.t. $\mathrm{q}_i \in \mathbf{q}_{answerable}$. Blue diamonds: question. Red stars: answer. Grey circles: context. (Q: "what is the quality of the chicken?", A: "The chicken was perfectly cooked and pretty tasty")

Answer vector agreements

As depicted and analyzed in the previous section, the model's hidden representations with respect to the answer span are clustered more closely in vector space for correct compared to wrong answer span predictions (see Figures 7.4, 7.6). This is particularly visible in the top three layers of the model, where generally high-level rather than low-level linguistic features are represented. To verify this observation more quantitatively, I've computed the average cosine similarities among all hidden representations for each token in the answer span whenever the correct answer contained more than one token. Hence, the following analysis was conducted exclusively for answerable questions since the correct answer span for unanswerable questions corresponds to the special `[CLS]` token.

Prior to the latter computation, I've removed all feature representations corresponding to the special `[PAD]` token and transformed the matrix of hidden representations $\mathbf{H}_i \in \mathbf{R}^{T \times D2}$ for each sentence pair sequence $(\mathbf{q}, \mathbf{c})_i$ into a lower-dimensional space to remove noise and exclusively keep those principal components that explain the most variance among the feature representations. In so doing, I've leveraged Principal Components Analysis (PCA) [72] and retained 95% of the hidden representations' variance. In initial experiments, I've experimented with retaining only 90% of the variance but results obtained from those transformations revealed less insightful analyses. Thus, I've proceeded with maintaining 95% of the features' variance throughout all analyses. This yielded a matrix of transformed hidden representations $\tilde{\mathbf{H}}_i \in \mathbf{R}^{T \times P}$, for each sentence pair $(\mathbf{q}, \mathbf{c})_i$. From the transformed matrix of hidden representations, I've extracted the matrix of hidden representations corresponding to answer span tokens $\tilde{\mathbf{H}}_{a(i)} \in \mathbf{R}^{T_a \times P}$ to compute the average cosine similarity solely across all answer vectors. The average cosine similarity among the rows (i.e., vectors) of the matrix $\tilde{\mathbf{H}}_{a(i)} \in \mathbf{R}^{T_a \times P}$ was computed as follows:

$$\cos_{a(i)} = \frac{1}{T_a T_a - T_a} \sum_{j}^{T_a} \sum_{k(k \neq j)}^{T_a} \cos(H^j_{a(i)}, H^k_{a(i)}) \in \mathbf{R}^P, \tag{7.1}$$

where the cosine similarity between two non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ is defined such as:

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{7.2}$$

The cosine similarity evaluates to a normalized dot product (i.e., dot product normalized by the magnitudes of the respective vectors) between two non-zero vectors and ranges from -1 (exactly opposite) to 1 (exactly the same), where 0 refers to orthogonal or decorrelated vector pairs. Hence, the cosine similarity between two vectors is always in the interval $[-1, 1]$. The closer the cosine similarity is to 1, the more similar two vectors are, and vice versa, the closer it is to -1, the more dissimilar the vector pair is. The latter computation (see Equation 7.1) was performed for the two sets of correct and erroneous answer span predictions separately to inspect potential differences between the two with respect to their average cosine similarities. This was done at each transformer layer $l \in L$, where L = 6, to examine shifts in the cosine similarity distributions across space.

Subjective questions As can be inferred from the probability density functions (PDFs) with respect to $\cos_a$ for the hidden representations of a model that was fine-tuned and evaluated on SubjQA (see Figure 7.7), the difference between the two PDFs with respect to $\cos_a$ is statistically significant for the layers 4, 5, and 6 with $p < .001$ according to an independent t-test. Bonferroni correction was applied to counteract the multiple comparisons problem, that is an increase in the likelihood of rare observations due to multiple statistical tests. Each observed p-value was multiplied by the number of tested hypotheses m, where $m = L = 6$. Hence, $p_{corrected} = p_{observed} \times m$. This p-value adjustment was applied to all subsequent statistical analyses.

Interestingly, the difference between PDFs with respect to $\cos_a$ for correct and wrong answer span predictions respectively at layer 1 also is statistically significant. This might, however, be an artifact rather than

[2] T is equal to the number of tokens in the word sequence $(\mathbf{q}, \mathbf{c})_i$ without appended `[PAD]` tokens and D = 768 which is the model's hidden size in each layer.

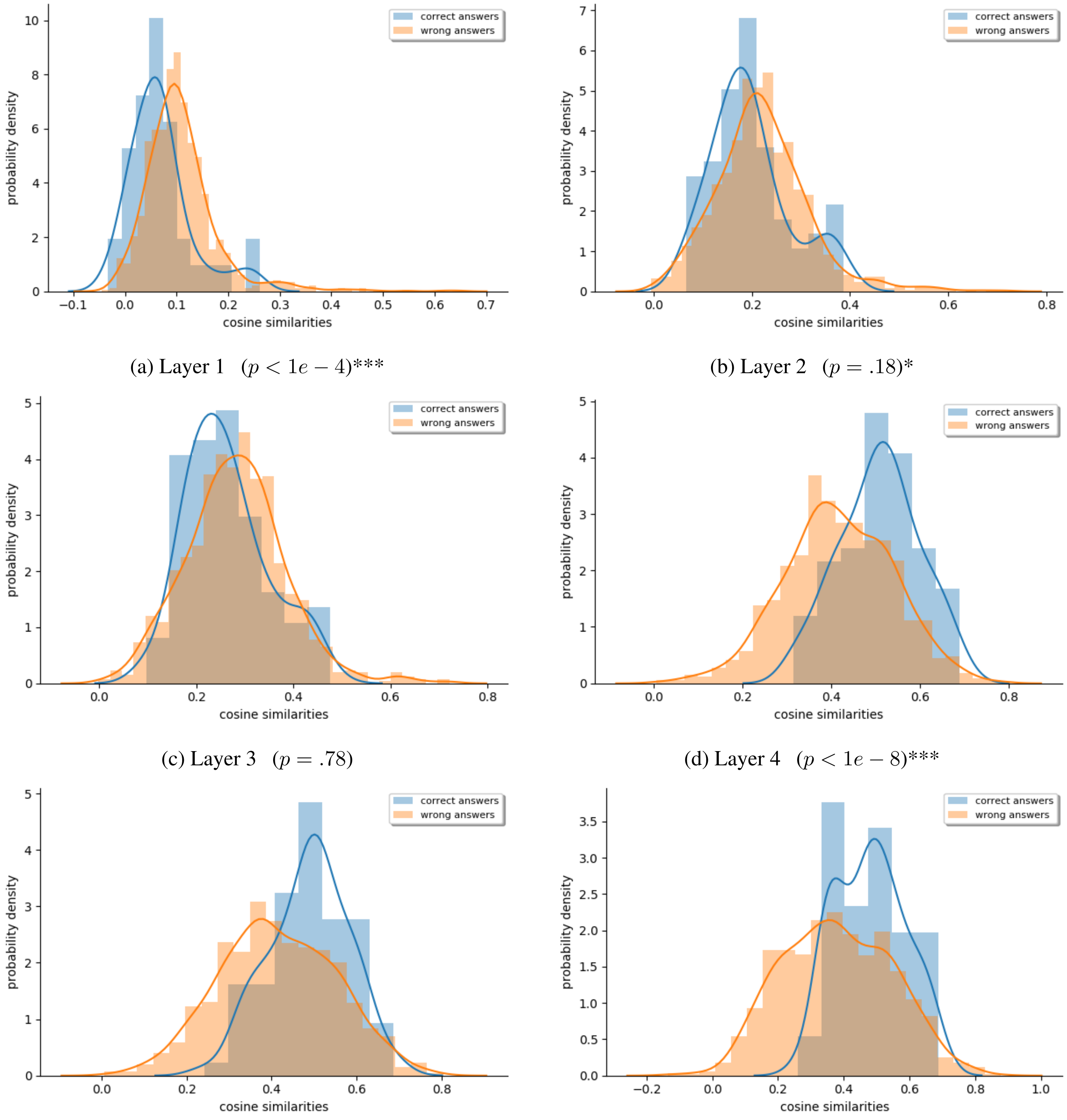
insightful information as layer 1 exclusively reflects low-level linguistic features, and answer tokens are not expected to be clustered closely in latent space (see Figures 7.4a, 7.5a, 7.6a). At both hidden layers 2 and 3, $\cos_a$ appears to be equally distributed for correct and erroneous answer predictions respectively. In layers 4, 5, and 6, however, $P(\cos_a > 0.5)$ is notably higher for correct compared to incorrect predictions (see Figure 7.7d, e, f). Hence, the probability of a high cosine similarity among elements of the answer vector matrix $\tilde{\mathbf{H}}_{a(i)} \in \mathbf{R}^{T_a \times P}$ is significantly larger for correct compared to erroneous answers. This is in line with the answer token clusters in 2D space (see Figures 7.4b-c, 7.5b-c, 7.6b-c).

Another key difference between the matrices $\tilde{\mathbf{H}}_{a(i)} \in \mathbf{R}^{T_a \times P}$ corresponding to correct and erroneous answer predictions respectively is the fact that probability mass with respect to $\cos_a$ constantly travels through space towards the right of the x-axis for the former but appears almost stagnant for the latter. This indicates that the likelihood for $\cos_a$ being close to 1 consistently increases through the layer hierarchy of the network whenever the model correctly predicted an answer span. In layer 1 most probability mass is distributed to $P(\cos_a < 0.1)$ (see Figure 7.7a), whereas in both the penultimate and last layer the PDF is centered around $P(0.4 < \cos_a < 0.7)$ (see Figure 7.7e, f). This pattern does not seem to be evident when the model did not get an answer span correct. The area of the PDFs corresponding to the latter scenario is spread in space more horizontally and hence does not clearly indicate a center.

If an answer span was predicted correctly, their hidden representations were most probably clustered closely in vector space in the hidden layers 4, 5, and 6, and vice versa, if the hidden representations with respect to the answer span were clustered closely in vector space, the model most probably predicted the correct start and end positions. This, however, is not always the case, as can be inferred from the PDFs. Sometimes $\cos_a$ is high although the model did not predict the correct answer span. Why this happens goes beyond the scope of this project and is encouraged to be inspected in future studies. The probability for the latter scenario is, however, fairly low. To provide an example: At hidden layer 5, $P(\cos_a > 0.5) \approx 0.64$ and $P(\cos_a > 0.5) \approx 0.29$ for answer vectors corresponding to correct and incorrect model predictions respectively (see Figure 7.7e).

The same pattern as depicted in the probability distributions is shown in the box plots (see Figure 7.8). The difference between $\bar{\cos}_a$ corresponding to correct and incorrect answer span predictions respectively is statistically significant at hidden layers 4, 5, and 6 with $p < .001$ according to both an independent *t*-test and a one-way ANOVA, and is higher for answer vectors corresponding to correct compared to incorrect predictions. Furthermore, the spread of $\cos_a$ values is notably higher for answer vectors corresponding to erroneous predictions as can be inferred from the whiskers of the box plots.

Objective questions I've performed the same computations as outlined above for the hidden representations of a model that was fine-tuned and evaluated on SQuAD (see Figures 7.9, 7.10) to investigate whether a different pattern holds for objective questions. The distributions with respect to $\cos_a$ are highly similar to the subjective case as can be inferred from both the PDFs (see Figure 7.9) and box plots (see Figure 7.10). The only difference between subjective and objective question - context pairs is that $\cos_a$ with respect to objective questions is significantly higher for answer vectors corresponding to correct compared to incorrect predictions in layers 5, 6 - compared to layers 4, 5 and 6 with respect to subjective questions. Again, differences in the mean cosine similarities between correct and erroneous model predictions are significant according to an independent *t*-test with $\alpha = .5$. p-values were adjusted following Bonferroni correction (see above). It seems as if the separation of objective answers from the context started later in latent space than the separation of subjective answers from its context, whenever the model correctly predicted start and end positions of an answer span (otherwise scarcely any separation is visible which holds for both types of questions).



(a) Layer 1 $(p < 1e-4)$***

(b) Layer 2 $(p = .18)$*

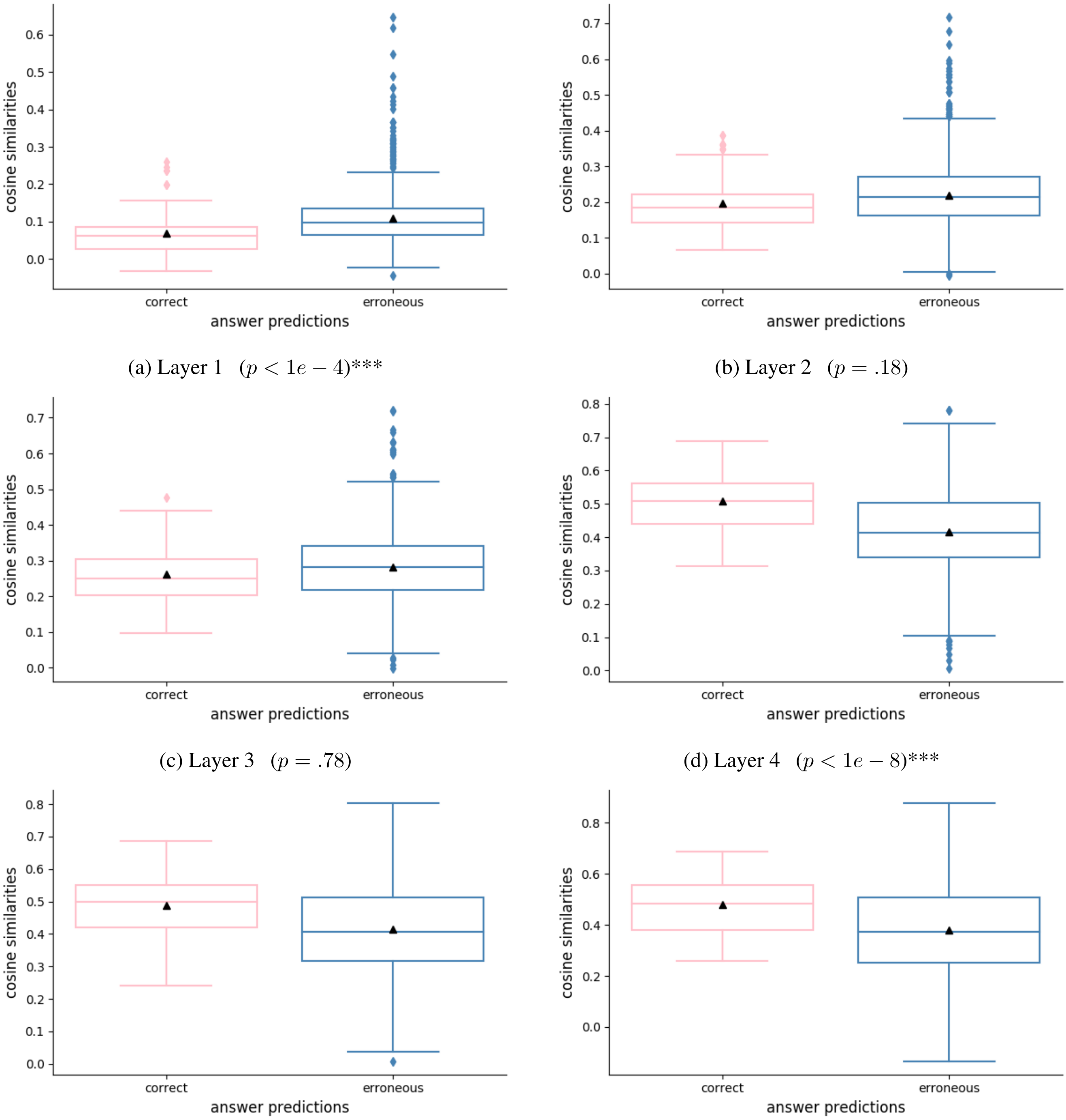
(c) Layer 3 $(p = .78)$

(d) Layer 4 $(p < 1e-8)$***

(e) Layer 5 $(p < 1e-5)$***

(f) Layer 6 $(p < 1e-6)$***

Figure 7.7: Probability density functions (PDFs) with respect to the average `cosine` similarities among hidden representations for each answer span token at each layer in the $D_{test} \in$ SubjQA, split into correct and erroneous model predictions. The respective model was fine-tuned on SubjQA. Cosine similarity distributions across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1^{st}) (top-left) and stopping at the last (6^{th}) (bottom-right) layer. Blue: correct answers. Orange: wrong answers. p-values to the right of each caption refer to the difference significance w.r.t. the mean `cosine` similarities according to independent t-tests ($p < .05$ = *, $p < .01$ = **, $p < .001$ = ***).



(a) Layer 1 $(p < 1e-4)$***

(b) Layer 2 $(p = .18)$

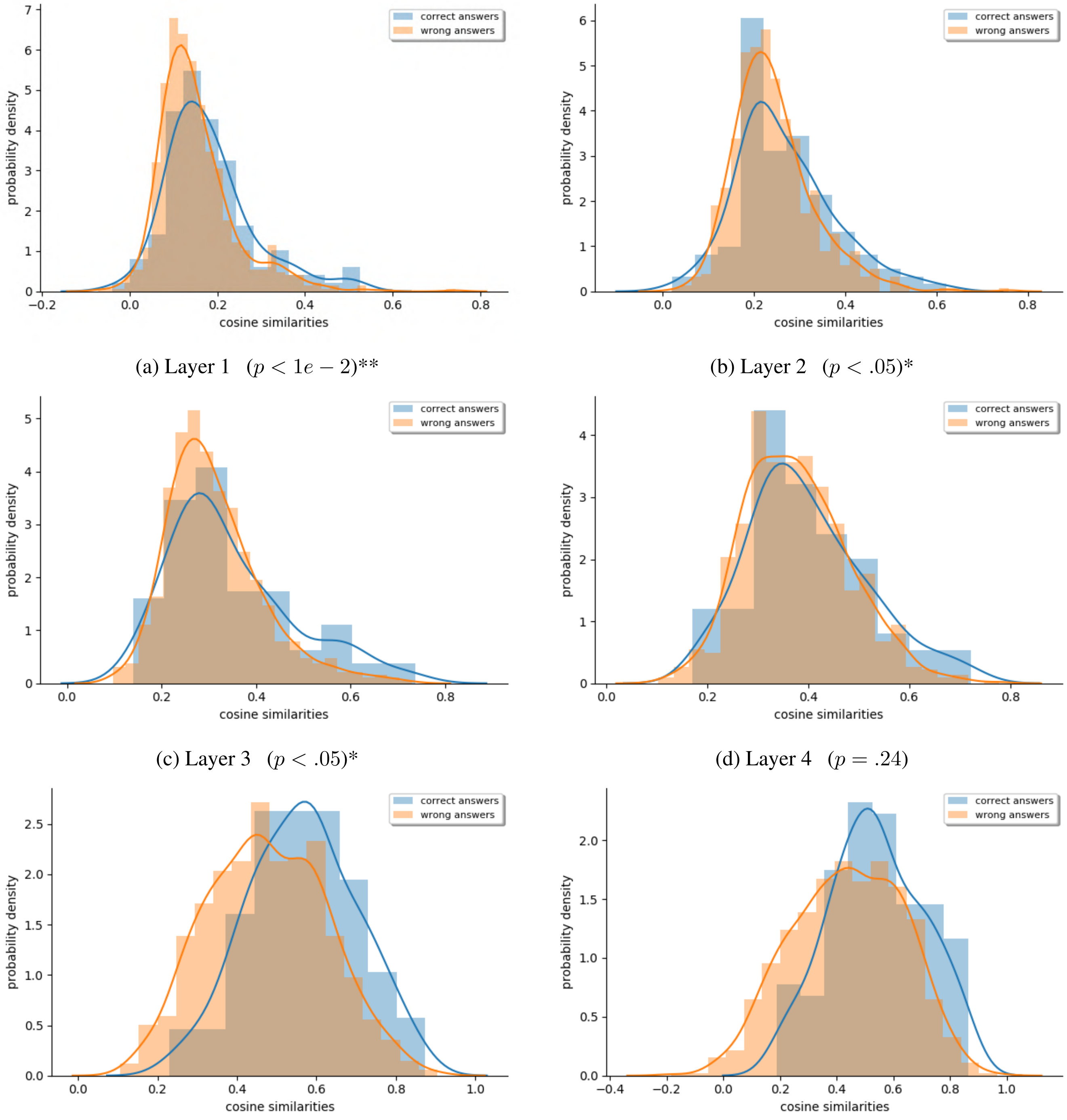
(c) Layer 3 $(p = .78)$

(d) Layer 4 $(p < 1e-8)$***

(e) Layer 5 $(p < 1e-5)$***

(f) Layer 6 $(p < 1e-6)$***

Figure 7.8: Box plots with respect to the average cosine similarities among hidden representations for each answer span token at each layer in D_{test} $\in$ SubjQA, split into correct and erroneous model predictions. The respective model was fine-tuned on SubjQA. Cosine similarity distributions across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1st) (top-left) and stopping at the last (6th) (bottom-right) layer. Pink: correct answers. Blue: wrong answers. $\triangle$: mean. p-values to the right of each caption refer to the difference significance w.r.t. the mean cosine similarities according to independent t-tests ($p < .05$ = *, $p < .01$ = **, $p < .001$ = ***).



(a) Layer 1 $(p < 1e-2)$**

(b) Layer 2 $(p < .05)$*

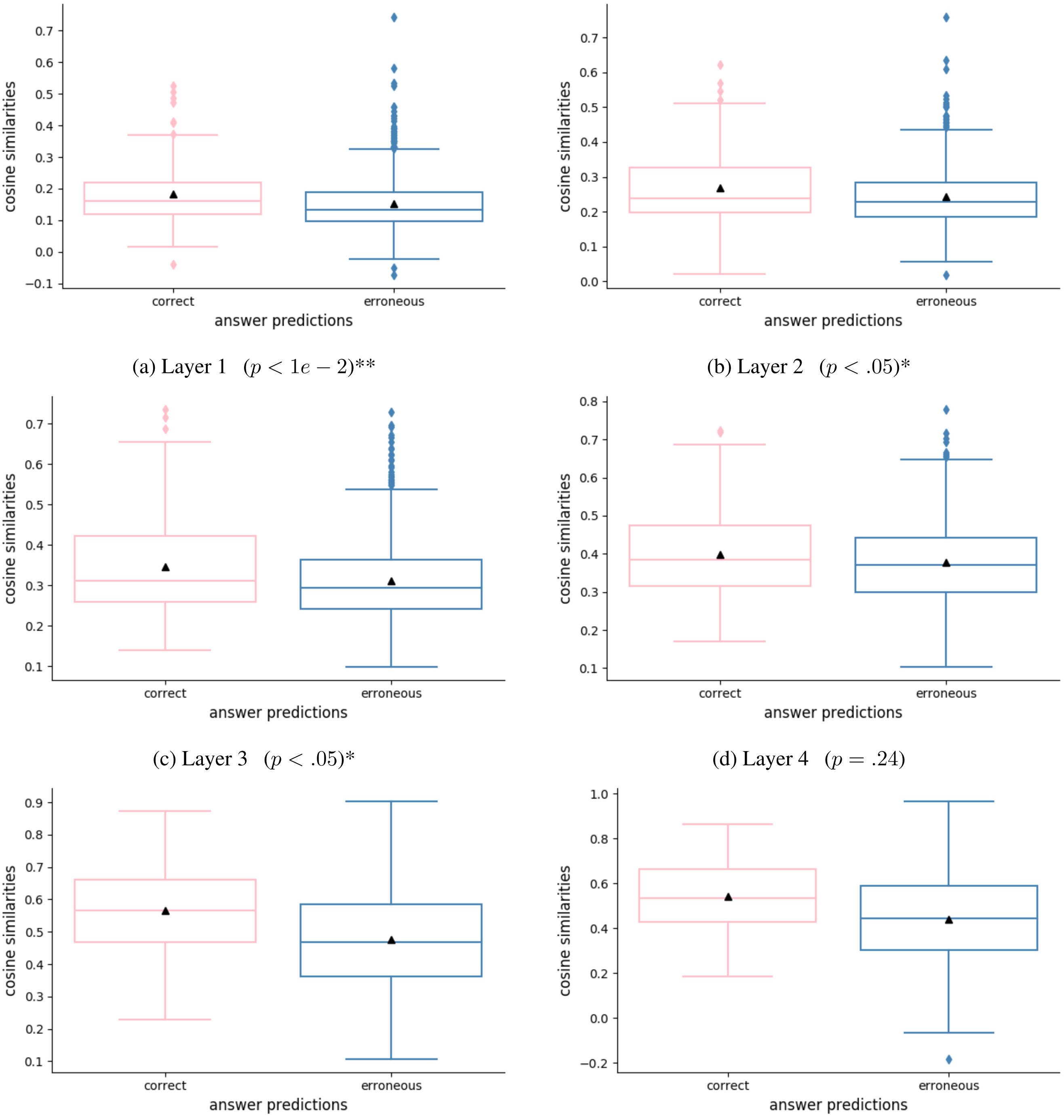
(c) Layer 3 $(p < .05)$*

(d) Layer 4 $(p = .24)$

(e) Layer 5 $(p < 1e-8)$***

(f) Layer 6 $(p < 1e-7)$***

Figure 7.9: Probability density functions (PDFs) with respect to the average `cosine` similarities among hidden representations for each answer span token at each layer in the $D_{test} \in$ SQuAD, split into correct and erroneous model predictions. The respective model was fine-tuned on SQuAD. Cosine similarity distributions across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1^{st}) (top-left) and stopping at the last (6^{th}) (bottom-right) layer. Blue: correct answers. Orange: wrong answers. p-values to the right of each caption refer to the difference significance w.r.t. the mean `cosine` similarities according to independent t-tests ($p < .05$ = *, $p < .01$ = **, $p < .001$ = ***).



(a) Layer 1 $(p < 1e - 2)$**

(b) Layer 2 $(p < .05)$*

(c) Layer 3 $(p < .05)$*

(d) Layer 4 $(p = .24)$

(e) Layer 5 $(p < 1e - 8)$***

(f) Layer 6 $(p < 1e - 7)$***

Figure 7.10: Box plots with respect to the average `cosine` similarities among hidden representations for each answer span token at each layer in $D_{test} \in$ SQuAD, split into correct and erroneous model predictions. The respective model was fine-tuned on SQuAD. Cosine similarity distributions across the different layers are represented from left-to-right and top-to-bottom in the usual bottom-up representational hierarchy starting at the first (1^{st}) (top-left) and stopping at the last (6^{th}) (bottom-right) layer. Pink: correct answers. Blue: wrong answers. $\triangle$: mean. p-values to the right of each caption refer to the difference significance w.r.t. the mean `cosine` similarities according to independent t-tests ($p < .05$ = *, $p < .01$ = **, $p < .001$ = ***).

Chapter 8

Discussion

In this section, I elaborate on both quantitative and qualitative results, discuss caveats and shortcomings with respect to experiments and analyses, and envision potential directions for future research.

8.1 General

In general, results have shown that it is crucial to fine-tune BERT on SubjQA to achieve state-of-the-art (SOTA) performance with respect to this dataset (see Table 6.1). A BERT model that was previously fine-tuned on SQuAD appears to not generalize well to SubjQA. This is not surprising, given that SQuAD is a span-selection QA dataset that exclusively contains objective questions with respect to a single domain, namely Wikipedia [62, 61]. In contrast, SubjQA consists of questions that ask for subjective opinions with respect to six domains (see Table 5.2) which requires different, or rather, additional abilities from a learner. This finding is in line with other work that emphasized the necessity of fine-tuning BERT on a specific dataset to achieve SOTA [21, 68, 23, 64].

8.2 Single-task Learning

Quantitative analyses revealed insights into the performance of different model architectures. In single-task learning (STL) settings, additional POST-BERT encoding layers yielded an increase in $F1$ over the baseline model, which represents the most common QA set-up, where a fully-connected linear layer follows a pre-trained BERT feature extractor [21, 68]. This is in line with one recent study that has shown that an additional encoding of input sequences through Highway or recurrent layers improves QA performance for SQuAD [37]. The best STL model in this work leveraged a BiLSTM [35] in-between the pre-trained BERT model and the task-specific linear output layer. Thus, one can draw the conclusion that the computation of temporal dependencies forward and backward in time yields an additional rise in performance when applied to BERT. This result is reasonable given the fact that temporal dependencies between tokens are not computed in Transformer based architectures [77, 21] (see Section 3.2) and have proven to unveil crucial information about the relationships between timesteps in temporal data [69, 70, 37] of which word sequences are a subset.

8.3 Multi-task Learning

Rather surprisingly, models that were trained in multi-task learning (MTL) settings could not significantly improve over models trained in STL set-ups. This finding is against results obtained from previous studies concerning MTL [4, 12, 9, 65]. Note, however, that none of these studies investigated neural architectures based on the Transformer [77] or examined QA behaviour. Recall that QA is a span-selection task whose nature is different from simple classification (i.e., sequence classification) and structured prediction (e.g., Part-of-Speech tagging, Named Entity Recognition) problems evaluated in the aforementioned studies. Although my best performing model was a learner that was adversarially trained on the binary task of subjectivity classification in an MTL set-up, the differences to STL are not significant and may thus be considered some sort of regularization yielded through the injection of noise. This most likely had multiple reasons.

Firstly, the binary classification task of labeling sentence pairs as reflecting a subjective opinion, or containing an objective, measurable fact has proven to be not solvable by any of the implemented architectures. None of the models achieved an $F1$-score $> 54\%$ which is scarcely better than tossing a coin (see Table 6.8). More detailed analyses concerning this auxiliary task revealed that the poor performances are primarily due to the inability of a learner to distinguish between objective and subjective questions in SubjQA (see Table 6.9 and Figure 6.4). Hence, evidence suggests that the human-provided labels either lack quality and require revision, or are too ambiguous - owing to the nature of subjectivity itself - in order to be utilized for a binary sequence classification task. SubjQA shall rather be considered an entirely subjective dataset. Both questions and answers might show varying degrees of subjectivity, but they cannot be classified under the umbrella of objectivity. The latter is not only encouraged by the quantitative results but even more so by qualitative analyses of the model's hidden representation in vector space.

The hidden representations of a model that was fine-tuned solely on subjectivity classification projected into $\mathbf{R}^2$ unveil that the model did not distinguish between objective and subjective questions with respect to SubjQA in latent space (see Figure 7.1). The model could, however, perfectly separate objective questions belonging to SQuAD from any question belonging to SubjQA. One potential caveat with the former analysis is that the model might have simply learned differences between the two datasets and not between subjectivity and objectivity, and therefore clustered both objective and subjective questions belonging to SubjQA in the same latent space but separated questions with respect to SQuAD. To alleviate the latter objection, I've projected the hidden states of an MTL QA model that was adversarially trained on the binary task of classifying input sequences into their respective datasets with PCA and t-SNE into $\mathbf{R}^2$. The synthetic labels were assigned to a sentence pair after inference solely for the purpose of visualization in 2D space. Even in this dataset agnostic set-up, the transformed hidden representations depict a clear separation of objective questions belonging to SQuAD from questions with respect to SubjQA but show no differentiation within SubjQA at any layer stage (see Figure 7.2). Hence, it appears reasonable to modify the binary task of subjectivity classification into a regression problem, where subjectivity is measured on a continuous scale. Whether this helps in an MTL setting is to be evaluated in future studies.

Secondly, although each of the implemented model architectures could easily solve the multi-way classification problem of labeling question - context sentence pairs with one of the six review domains (see Table 6.10) it seems as if knowledge about the latter task either maintained or deteriorated rather than enhanced performance (see Table 6.4). Adversarial training with respect to context-domain classification in an MTL set-up made the model perform slightly better than without forcing the model to learn domain-invariant features, but did not help either. This might indicate that a Transformer based architecture such as BERT [21] encodes domain-invariant features by itself without the necessity of an additional adversarially trained auxiliary task. The latter conclusion is in line with recent research about BERT [64], but requires more investigation into the feature representations of BERT and its ability to generalize across domains. This is an avenue of research I highly encourage to pursue as it will yield apprehension of both MTL and adversarial training for BERT in particular and Transformers in general.

8.4 Interrogative Words & Review Domains

Questions that start with `how` were the most difficult to answer (see Table 6.7) despite appearing more frequently in the train set of SubjQA than questions starting with any other interrogative word (see Table 5.5). This indicates that a considerable number of subjective questions begins with `how`, and subjective questions are difficult to answer in general. Why QA performance with respect to the domain `tripadvisor` was significantly worse compared to model performances across other domains is yet to be deciphered. This is another avenue I leave for future research that is concerned with SubjQA. An intuitive explanation, however, might be the fact that `tripadvisor` appeared considerably more often in the test set of SubjQA than any other domain, while being the least frequently encountered review domain during training (see Table 5.3).

8.5 Hidden Representations

Fine-grained error analyses unveiled that hidden representations in top layers with respect to the correct answer tokens are clustered closely in vector space and separated from the context for correct answer span prediction, and vice versa neither clustered together nor separated from the context for erroneous predictions (see Figures 7.4, 7.5, 7.6). The latter insights hold for both subjective and objective questions. These qualitative results are in line with one recently published study that has conducted a similar analysis of BERT's hidden representations in vector space with respect to SQuAD [2].

8.6 Cosine Similarity Distributions

To the best of my knowledge, I am, however, the first to date who has investigated further into BERT's hidden representations with respect to both correct and erroneous answer predictions. I have analyzed the cosine similarities among answer token representations at every layer stage to both decipher why the aforementioned differences in latent space between the two prediction sets occur and provide a mathematical formalization to quantitatively verify the results obtained from the qualitative analyses (see Section 7.5.1). Without the latter, one could not draw general conclusions. Probability density functions (PDFs) and box-plots displayed that there are no statistically significant differences between the two prediction sets with respect to their average cosine similarity among answer token hidden representations, $\cos_a$, (see Equation 7.2 for the computation of $\cos_a$), at early Transformer layers, namely layers 1, 2, and 3. There are, however, statistically significant differences between $\bar{\cos}_a$ corresponding to correct and erroneous answer predictions respectively at later Transformer layers, namely layers 4, 5, and 6 (see Figures 7.7, 7.8, 7.9, 7.10). Note that these results reflect correlation rather than causation. As can be inferred from the PDFs (e.g., Figure 7.7), $P(\cos_a > .4)$ in layers 4, 5, and 6 is higher for correct answer predictions compared to erroneous predictions, and vice versa, the probability for a correct model prediction is greater when $\cos_a$ is high in top layers of the model. Hence, one can draw the conclusion that $\cos_a$ and $P(\hat{y} = y)$ are positively correlated.

If one leverages the knowledge about similarities of answer token representations in top layers of a Transformer model to anticipate whether an answer span prediction will be correct or erroneous, an erroneous answer could simply be skipped without the necessity to inspect its validity a posteriori. In a follow-up study that was performed alongside this master's thesis, we show that this insight has decisive implications for down-stream applications. In Muttenthaler et al. [53], we propose an unsupervised evaluation method that can almost faultlessly predict the correctness of an answer span prediction, without the need for any labeled data at inference time. This method might be applied to semi-automatic generation of QA datasets, where a predicted answer span could be considered as gold label, if it was identified as correct by our method. Hence, the need for tedious annotation work could notably be reduced.

Moreover, instead of exclusively utilizing cross-entropy loss with respect to the model's output logits to train a neural network for QA, one could additionally exploit a different loss function (e.g., cosine-embedding loss) that aims at increasing the similarity between token representations with respect to the correct answer span in top layers of the model (e.g., in the penultimate and last Transformer layer). I plan to investigate further into this line of research in follow-up work.

8.7 Conclusions

To summarize, the main conclusions that can be drawn from this study are as follows.

1. Fine-tuning on a dataset that contains subjective questions is indispensable to answer subjective questions since objective questions do not appear to generalize to the realm of subjectivity. This answers **RQ** 1.

2. Encoding temporal dependencies among tokens forward and backward in time as a POST-BERT encoding step before performing QA enhances performance. This reveals insights about **RQ** 2.

3. Multi-task learning (MTL) does not appear to particularly enhance BERT's answering behavior. This is in part due to the impossibility of solving one of the two auxiliary tasks that were leveraged in this study. Hence, SubjQA shall be considered an entirely subjective dataset or a dataset with varying degrees of subjectivity that does not contain objective questions. This is most likely due to the ambiguity of crowd-sourced labels. I, therefore, encourage the measuring of subjectivity on a continuous scale and modifying of the binary classification task into a multi-class or regression problem accordingly. This unveils additional information that was not considered an area of examination prior to conducting experiments. Thus, **RQ** 3 remains open to investigation with the exploitation of different auxiliary tasks, improved labels, and modified subjectivity measurements.

4. Questions that start with `how` or belong to the domain `tripadvisor` are the most difficult to solve in SubjQA. This corresponds to both **RQ** 4 & 5.

5. A model's hidden representations with respect to the correct answer tokens are clustered more closely in low-dimensional vector space in top Transformer layers than answer token representations corresponding to erroneous predictions. With this insight, **RQ** 6 can be regarded as investigated.

6. The probability to achieve a high cosine similarity (e.g., $> .5$) among the answer token vectors in latent space is significantly greater for correct compared to erroneous predictions. This may be considered an additional analysis that appeared insightful while scrutinizing **RQ** 6. To the best of my knowledge, I am the first to date who has ever investigated this. In Muttenthaler et al. [53], we have shown that this information can easily be leveraged for down-stream applications to predict correctness of an answer span prediction in Transformer-based models across two datasets and seven domains. Further research is encouraged to examine whether this holds across other QA datasets and domains.

Chapter 9

Summary

First, I provided an **Overview** of the sections in this thesis. Second, I introduced the overarching **Topic** of the thesis, namely QA with respect to subjective natural language questions, motivated the research goals and explained the general task of **QA**. In so doing, I explored various neural network architectures such as the **Transformer**, which served as the foundation for each implemented model, **RNNs**, and **Highway Networks**, all of which were exploited in the various experiments to encode token sequences differently and determine potential benefits. Third, I mentioned **Research** that is related to my analyses. Fourth, I first explained the mechanisms and conceptual details behind the leveraged models, namely BERT, RNNs, Highway networks, and later in Section 4 discussed the fine-tuning procedure, and most crucially clarified how the networks are optimized with respect to the particular tasks. Fifth, I provided a thorough analysis of the utilized **Datasets**, discussed both characteristics of and differences between the two. Sixth, in Section 6, I presented and explained results concerning all conducted experiments.The seventh section regarding **Qualitative Analyses** aimed at deciphering the network's feature representations in latent space with respect to both classes and individual sentence pairs. In so doing, I analyzed `why` and `where` along the way (i.e., in vector space) the selected model made mistakes in its predictions to investigate deeper into the model's errors. Finally, I **discussed** the results obtained from both quantitative and qualitative analyses as thoroughly as space and time allowed, drew conclusions and envisioned potential directions for future research.

Chapter 10

Acknowledgments

This thesis was almost entirely written during the Covid-19 crisis. A few months back in time, I would have never expected to write my master's thesis mainly at my mother's apartment back in my hometown Vienna. As ever, life holds surprises, not even the human brain - which is considered to be the holy grail of intelligence - is able to foresee. I am incredibly grateful that my mother hosted me during this precarious time, and provided a desk on which I could conduct experiments, analyze data, inspect algorithms and let creativity flow as much as space and constrained conditions allowed. Special thanks to my wonderful supervisor, Johannes Bjerva, who has always supplied constructive feedback and listened to me when I needed it the most. Alexander Haseler and Nora Hollenstein have provided particularly valuable feedback on earlier drafts of this master's thesis, which definitely contributed to improved reading clarity. I am grateful that they took the time to read over my work. I would also like to thank my three closest friends, Julian Blumenschein, Tobias Kovats and Alicja Grządziel, who have provided lots of mental support during the lockdown. I don't know what my mind would have done without them. However, I have been lucky enough that the Covid-19 pandemic has neither affected me physically nor with respect to my future job prospects. Unfortunately, many people have lost their jobs, suffered from the virus, or even witnessed their beloved ones pass away. This thesis is dedicated to each of these people. We, as scientists, must draw attention to the ones who suffer, and strive for knowledge to alleviate their burdens. This might sound idealistic, but this is most definitely one of the reasons why I have decided to pursue the path of science. I hope this piece of research contributed to this pursuit of knowledge, even if just a little. Nevertheless, I am grateful that I could work on a fulfilling, truly satisfying, and sometimes mind-bending project throughout the pandemic.

Bibliography

[1] Domain adaptation in question answering. CoRR **abs/1702.02171** (2017), http://arxiv.org/abs/1702.02171, withdrawn.

[2] van Aken, B., Winter, B., Löser, A., Gers, F.A.: How does bert answer questions? a layer-wise analysis of transformer representations. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1823—-1832. CIKM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3357384.3358028, https://doi.org/10.1145/3357384.3358028

[3] Allen, I.E., Seaman, C.A.: Likert scales and data analyses. Quality progress **40**(7), 64–65 (2007)

[4] Alonso, H.M., Plank, B.: When is multitask learning effective? semantic sequence prediction under varying data conditions. In: Lapata, M., Blunsom, P., Koller, A. (eds.) Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers. pp. 44–53. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/e17-1005, https://doi.org/10.18653/v1/e17-1005

[5] Arkhangelskaia, E., Dutta, S.: Whatcha lookin' at? deeplifting bert's attention in question answering. CoRR **abs/1910.06431** (2019), http://arxiv.org/abs/1910.06431

[6] Ba, L.J., Kiros, J.R., Hinton, G.E.: Layer normalization. CoRR **abs/1607.06450** (2016), http://arxiv.org/abs/1607.06450

[7] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.0473

[8] Banfield, A.: Unspeakable Sentences: Narration and Representation in the Language of Fiction. Boston: Routledge and Kegan Paul (1982)

[9] Bingel, J., Søgaard, A.: Identifying beneficial task relations for multi-task learning in deep neural networks. In: Lapata, M., Blunsom, P., Koller, A. (eds.) Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers. pp. 164–169. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/e17-2026, https://doi.org/10.18653/v1/e17-2026

[10] Biswas, P., Sharan, A., Malik, N.: A framework for restricted domain question answering system. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). pp. 613–620 (Feb 2014). https://doi.org/10.1109/ICICICT.2014.6781351

[11] Bjerva, J.: One model to rule them all: Multitask and multilingual modelling for lexical analysis. CoRR **abs/1711.01100** (2017), http://arxiv.org/abs/1711.01100

[12] Bjerva, J.: Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In: Tiedemann, J., Tahmasebi, N. (eds.) Proceedings of the 21st Nordic Conference on Computational Linguistics, NODALIDA 2017, Gothenburg, Sweden, May 22-24, 2017. Linköping Electronic Conference Proceedings, vol. 131, pp. 216–220. Linköping University Electronic Press / Association for Computational Linguistics (2017), http://www.ep.liu.se/ecp/article.asp?issue=131&article=025&volume=

[13] Bjerva, J., Bhutani, N., Golshan, B., Tan, W., Augenstein, I.: Subjqa: A dataset for subjectivity and review comprehension. CoRR **abs/2004.14283** (2020), https://arxiv.org/abs/2004.14283

[14] Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Wang, J.T. (ed.) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008. pp. 1247–1250. ACM (2008). https://doi.org/10.1145/1376616.1376746, https://doi.org/10.1145/1376616.1376746

[15] Cabrio, E., Cojan, J., Aprosio, A.P., Magnini, B., Lavelli, A., Gandon, F.: Qakis: an open domain QA system based on relational patterns. In: Glimm, B., Huynh, D. (eds.) Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA, November 11-15, 2012. CEUR Workshop Proceedings, vol. 914. CEUR-WS.org (2012), http://ceur-ws.org/Vol-914/paper_24.pdf

[16] Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997). https://doi.org/10.1023/A:1007379606734, https://doi.org/10.1023/A:1007379606734

[17] Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. In: Barzilay, R., Kan, M. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. pp. 1870–1879. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-1171, https://doi.org/10.18653/v1/P17-1171

[18] Cheng, X., Xu, W., Chen, K., Wang, W., Bi, B., Yan, M., Wu, C., Si, L., Chu, W., Wang, T.: Symmetric regularization based BERT for pair-wise semantic reasoning. CoRR **abs/1909.03405** (2019), http://arxiv.org/abs/1909.03405

[19] Chopra, S., Balakrishnan, S., Gopalan, R.: Dlid: Deep learning for domain adaptation by interpolating between domains. In: ICML workshop on challenges in representation learning. vol. 2 (2013)

[20] Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR **abs/1412.3555** (2014), http://arxiv.org/abs/1412.3555

[21] Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423, https://doi.org/10.18653/v1/n19-1423

[22] Doan-Nguyen, H., Kosseim, L.: Improving the precision of a closed-domain question-answering system with semantic information. In: Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval. p. 850–859. RIAO '04, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, Paris, FRA (2004)

[23] Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., Smith, N.A.: Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. CoRR **abs/2002.06305** (2020), https://arxiv.org/abs/2002.06305

[24] Dosovitskiy, A., Springenberg, J.T., Brox, T.: Learning to generate chairs with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1538–1546. IEEE Computer Society (2015). https://doi.org/10.1109/CVPR.2015.7298761, https://doi.org/10.1109/CVPR.2015.7298761

[25] Downey, A.R.O.K.M., Rumshisky, A.: Getting closer to ai complete question answering: A set of prerequisite real tasks (2020), https://www.aaai.org/Papers/AAAI/2020GB/AAAI-RogersA.7778.pdf

[26] Elman, J.L.: Finding structure in time. Cognitive Science **14**(2), 179–211 (1990). https://doi.org/10.1207/s15516709cog1402_1, https://doi.org/10.1207/s15516709cog1402_1

[27] Ganin, Y., Lempitsky, V.S.: Unsupervised domain adaptation by backpropagation. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 1180–1189. JMLR.org (2015), http://proceedings.mlr.press/v37/ganin15.html

[28] Glass, M.R., Gliozzo, A., Chakravarti, R., Ferritto, A., Pan, L., Bhargav, G.P.S., Garg, D., Sil, A.: Span selection pre-training for question answering. CoRR **abs/1909.04120** (2019), http://arxiv.org/abs/1909.04120

[29] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, D.M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010. JMLR Proceedings, vol. 9, pp. 249–256. JMLR.org (2010), http://proceedings.mlr.press/v9/glorot10a.html

[30] Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011. pp. 513–520. Omnipress (2011), https://icml.cc/2011/papers/342_icmlpaper.pdf

[31] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90, https://doi.org/10.1109/CVPR.2016.90

[32] Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 1693–1701. Curran Associates, Inc. (2015), http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf

[33] Hewlett, D., Lacoste, A., Jones, L., Polosukhin, I., Fandrianto, A., Han, J., Kelcey, M., Berthelot, D.: Wikireading: A novel large-scale language understanding task over wikipedia. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers. The Association for Computer Linguistics (2016). https://doi.org/10.18653/v1/p16-1145, https://doi.org/10.18653/v1/p16-1145

[34] Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Honkela, T., Duch, W., Girolami, M.A., Kaski, S. (eds.) Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I.

Lecture Notes in Computer Science, vol. 6791, pp. 44–51. Springer (2011). https://doi.org/10.1007/978-3-642-21735-7_6, https://doi.org/10.1007/978-3-642-21735-7_6

[35] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735

[36] Howard, J., Ruder, S.: Fine-tuned language models for text classification. CoRR **abs/1801.06146** (2018), http://arxiv.org/abs/1801.06146

[37] Hu, Z.: Question answering on squad with bert. CS224N Report, Stanford University. Accessed pp. 01–09 (2019), https://pdfs.semanticscholar.org/91bc/cbf3875b5f000e5469ac468db5569eabbc39.pdf

[38] Irsoy, O., Cardie, C.: Opinion mining with deep recurrent neural networks. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 720–728. ACL (2014). https://doi.org/10.3115/v1/d14-1080, https://doi.org/10.3115/v1/d14-1080

[39] Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 495–502. Omnipress (2010), https://icml.cc/Conferences/2010/papers/100.pdf

[40] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980

[41] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 1106–1114 (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[42] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: A lite BERT for self-supervised learning of language representations. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=H1eA7AEtvS

[43] LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539, https://doi.org/10.1038/nature14539

[44] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Forsyth, D.A., Mundy, J.L., Gesù, V.D., Cipolla, R. (eds.) Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol. 1681, p. 319. Springer (1999). https://doi.org/10.1007/3-540-46805-6_19, https://doi.org/10.1007/3-540-46805-6_19

[45] Li, L., Jin, L., Huang, D.: Exploring recurrent neural networks to detect named entities from biomedical text. In: Sun, M., Liu, Z., Zhang, M., Liu, Y. (eds.) Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data - 14th China National Conference, CCL 2015 and Third International Symposium, NLP-NABD 2015, Guangzhou,

China, November 13-14, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9427, pp. 279–290. Springer (2015). https://doi.org/10.1007/978-3-319-25816-4_23, https://doi.org/10.1007/978-3-319-25816-4_23

[46] Liu, P., Joty, S.R., Meng, H.M.: Fine-grained opinion mining with recurrent neural networks and word embeddings. In: Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., Marton, Y. (eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. pp. 1433–1443. The Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/d15-1168, https://doi.org/10.18653/v1/d15-1168

[47] Liu, X., He, P., Chen, W., Gao, J.: Multi-task deep neural networks for natural language understanding. CoRR **abs/1901.11504** (2019), http://arxiv.org/abs/1901.11504

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

[49] Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 97–105. JMLR.org (2015), http://proceedings.mlr.press/v37/long15.html

[50] Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., Marton, Y. (eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. pp. 1412–1421. The Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/d15-1166, https://doi.org/10.18653/v1/d15-1166

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

[52] Mihalcea, R., Banea, C., Wiebe, J.: Multilingual subjectivity and sentiment analysis. In: The 50th Annual Meeting of the Association for Computational Linguistics, Tutorial Abstracts, July 8, 2012, Jeju Island, Korea. p. 4. The Association for Computer Linguistics (2012), https://www.aclweb.org/anthology/P12-4004/

[53] Muttenthaler, L., Augenstein, I., Bjerva, J.: Unsupervised evaluation for question answering. Under Review for EMNLP 2020 (2020)

[54] 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., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[55] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011), http://dl.acm.org/citation.cfm?id=2078195

[56] Peters, M.E., Ammar, W., Bhagavatula, C., Power, R.: Semi-supervised sequence tagging with bidirectional language models. In: Barzilay, R., Kan, M. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. pp. 1756–1765. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-1161, `https://doi.org/10.18653/v1/P17-1161`

[57] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Walker, M.A., Ji, H., Stent, A. (eds.) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/n18-1202, `https://doi.org/10.18653/v1/n18-1202`

[58] Phang, J., Févry, T., Bowman, S.R.: Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. CoRR **abs/1811.01088** (2018), `http://arxiv.org/abs/1811.01088`

[59] Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A comprehensive grammar of the English language. Institution: Longman (1985)

[60] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding with unsupervised learning. Technical report, OpenAI (2018)

[61] Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for squad. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers. pp. 784–789. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/P18-2124, `https://www.aclweb.org/anthology/P18-2124/`

[62] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100, 000+ questions for machine comprehension of text. In: Su, J., Carreras, X., Duh, K. (eds.) Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016. pp. 2383–2392. The Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/d16-1264, `https://doi.org/10.18653/v1/d16-1264`

[63] Ratnaparkhi, A.: A simple introduction to maximum entropy models for natural language processing. IRCS Technical Reports Series p. 81 (1997)

[64] Rogers, A., Kovaleva, O., Rumshisky, A.: A primer in bertology: What we know about how BERT works. CoRR **abs/2002.12327** (2020), `https://arxiv.org/abs/2002.12327`

[65] Ruder, S.: An overview of multi-task learning in deep neural networks. CoRR **abs/1706.05098** (2017), `http://arxiv.org/abs/1706.05098`

[66] Rush, A.: The annotated transformer. In: Proceedings of Workshop for NLP Open Source Software (NLP-OSS). pp. 52–60. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/W18-2509, `https://www.aclweb.org/anthology/W18-2509`

[67] Ryu, P., Jang, M., Kim, H.: Open domain question answering using wikipedia-based knowledge model. Inf. Process. Manag. **50**(5), 683–692 (2014). https://doi.org/10.1016/j.ipm.2014.04.007, `https://doi.org/10.1016/j.ipm.2014.04.007`

[68] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR **abs/1910.01108** (2019), `http://arxiv.org/abs/1910.01108`

[69] Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997). https://doi.org/10.1109/78.650093, `https://doi.org/10.1109/78.650093`

[70] Seo, M.J., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. CoRR **abs/1611.01603** (2016), `http://arxiv.org/abs/1611.01603`

[71] Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012. pp. 3288–3291. IEEE Computer Society (2012), `http://ieeexplore.ieee.org/document/6460867/`

[72] Shlens, J.: A tutorial on principal component analysis. CoRR **abs/1404.1100** (2014), `http://arxiv.org/abs/1404.1100`

[73] Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. CoRR **abs/1505.00387** (2015), `http://arxiv.org/abs/1505.00387`

[74] Sugawara, S., Stenetorp, P., Inui, K., Aizawa, A.: Assessing the benchmarking capacity of machine reading comprehension datasets. CoRR **abs/1911.09241** (2019), `http://arxiv.org/abs/1911.09241`

[75] Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., Suleman, K.: Newsqa: A machine comprehension dataset. CoRR **abs/1611.09830** (2016), `http://arxiv.org/abs/1611.09830`

[76] Vargas-Vera, M., Lytras, M.D.: Aqua: A closed-domain question answering system. Information Systems Management **27**(3), 217–225 (2010). https://doi.org/10.1080/10580530.2010.493825, `https://doi.org/10.1080/10580530.2010.493825`

[77] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), `http://papers.nips.cc/paper/7181-attention-is-all-you-need`

[78] Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Superglue: A stickier benchmark for general-purpose language understanding systems. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. pp. 3261–3275 (2019), `http://papers.nips.cc/paper/8589-superglue-a-stickier-benchmark-for-general-purpose-language-understanding-`

[79] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A multi-task benchmark and analysis platform for natural language understanding. CoRR **abs/1804.07461** (2018), `http://arxiv.org/abs/1804.07461`

[80] Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., Jiang, J.: R^3: Reinforced ranker-reader for open-domain question answering. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 5981–5988. AAAI Press (2018), `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16712`

[81] Weissenborn, D., Minervini, P., Dettmers, T., Augenstein, I., Welbl, J., Rocktäschel, T., Bosnjak, M., Mitchell, J., Demeester, T., Stenetorp, P., Riedel, S.: Jack the reader - A machine reading framework. CoRR **abs/1806.08727** (2018), http://arxiv.org/abs/1806.08727

[82] Weissenborn, D., Wiese, G., Seiffe, L.: Fastqa: A simple and efficient neural architecture for question answering. CoRR **abs/1703.04816** (2017), http://arxiv.org/abs/1703.04816

[83] Wiebe, J.: Learning subjective adjectives from corpora. In: Kautz, H.A., Porter, B.W. (eds.) Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. pp. 735–740. AAAI Press / The MIT Press (2000), http://www.aaai.org/Library/AAAI/2000/aaai00-113.php

[84] Wiebe, J., Wilson, T., Cardie, C.: Annotating expressions of opinions and emotions in language. Lang. Resour. Evaluation **39**(2-3), 165–210 (2005). https://doi.org/10.1007/s10579-005-7880-9, https://doi.org/10.1007/s10579-005-7880-9

[85] Yang, Y., Yih, W.t., Meek, C.: WikiQA: A challenge dataset for open-domain question answering. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2013–2018. Association for Computational Linguistics, Lisbon, Portugal (2015). https://doi.org/10.18653/v1/D15-1237, https://www.aclweb.org/anthology/D15-1237

[86] Yu, T., Li, Z., Zhang, Z., Zhang, R., Radev, D.R.: Typesql: Knowledge-based type-aware neural text-to-sql generation. In: Walker, M.A., Ji, H., Stent, A. (eds.) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers). pp. 588–594. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/n18-2093, https://doi.org/10.18653/v1/n18-2093

[87] Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., Zhou, X.: Semantics-aware BERT for language understanding. CoRR **abs/1909.02209** (2019), http://arxiv.org/abs/1909.02209

[88] Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. CoRR **abs/1709.00103** (2017), http://arxiv.org/abs/1709.00103