Multitask & Meta Learning for Language Models

An investigation of optimal multitask schemas and generalisation for the automatic construction of Knowledge Graphs with focus on sentiment

Ryan Jenkinson[1]

Computational Statistics and Machine Learning MSc

Industry Supervisors: Dr Erik Mathiesen, Dr Ryan Garland
Academic Supervisors: Prof. David Barber, Yap Pau Ching

Submission date: **Day** September 2019

To my parents, for their unwavering support in my ambitions.

Acknowledgements

Firstly, I'd like to thank the Machine Learning and Data Science team: Erik, Ryan, Sophia, James and Marina at Streetbees, for their continual encouragement and support throughout this thesis, and for adopting me as part of the team throughout the summer. Additional thanks go to Jeunghyun Byun, a fellow MSc student in Machine Learning, for his friendliness. Our discussions and meetings have given me valuable insight, and the emphasis on learning and development has been amazing. Secondly to Stanley, for his diligence and perseverance in helping me run my experiments. Thirdly, I'd like to thank my academic supervisor Yap Pau Ching, for her helpful discussions and keeping me on the right track. Finally, to my parents and family, my friends and long term partner, for their unwavering and continual support in all of my ambitions.

Abstract

In this thesis, we aim to investigate how multitask learning can affect the performance of Language Models on difficult downstream aspect based sentiment analysis tasks. We introduce novel task sampling schemas, and leverage dependent subtask hierarchies to fine tune our model jointly on several related tasks in order to regularise the learned embeddings and make them better generalisable to our main task. We report major improvements to the current state-of-the-art metrics, of over 2% relative increase in some cases, showcasing the efficacy of these schemes.

Additionally, we propose a full investigation into how different language models learn, and in particular the importance of how the pretraining methodology can lead to representations which generalise better to complicated downstream tasks such as this.

In order to generalise to new categories, we propose a novel application of Meta Learning to a Few Shot Text Classification regime, and outline how this could be implemented.

Contents

List of Tables

List of Figures

List of Common Terms and Symbols

This list outlines several symbols used frequently and consistently during this thesis.

General

$\mathcal{A}$	A set of aspect categories for the (T)ABSA tasks
$\mathcal{D}$	A dataset
$\mathbb{P}$	A probability distribution
(T)ABSA	(Target) Aspect Based Sentiment Analysis
document	A general term referring to a piece of text, which may be a single sentence, word or paragraph.
GLUE	General Language Understanding Evaluation benchmark - a common set of tasks for which we train, evaluate and analyse NLU systems
LM	Language Model
LSTM	Long Short Term Memory (Network/Architecture)
ML	Machine Learning
NER	Named Entity Recognition
NLI	Natural Language Inference - e.g similarity of two sentences or next sentence prediction task
NLP	Natural Language Processing
NLU	Natural Language Understanding
PoS	Part of Speech
QA	Question Answering
RNN	Recurrent Neural Network
SOTA	State of the Art

Support We adopt the mathematical definition of support: the number of nonzero points in a domain. In our case, it will refer to the number of instances of examples.

Language Modelling

BERT Bidirectional Encoder Representation using Transformers

ELMo Embeddings from Language Models

GPT Generalised Pre Training

RoBERTa Robustly optimised BERT approach

Multitask Learning

$\mathcal{T}$ A set of tasks, typically $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ where τ_i is task i.

$p(\mathcal{T})$ Probability distribution over our tasks. This is our notation for our sampling schema/distribution.

w_{τ_i} The weight of task $\tau_i \in \mathcal{T}$. The sampling distributions are proportional to this weight.

Meta Learning

$\mathcal{Q}$ Query Set

$\mathcal{S}$ Support Set

$\mathcal{T}$ A set of tasks, typically $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ where τ_i is task i.

Chapter 1

Introduction

In this chapter, we give an introduction of the project (including a brief history of the field), describe the relevancy of the problem and the specific research focus of this thesis, and give an overview of the thesis structure. Furthermore, we provide reference to a public repository, where detailed code accompanying this project can be located. While many of the ideas in this project can be generalised to other subfields of Natural Language Processing (NLP), we focus in particular on Sentiment Analysis, and specifically on Aspect Based Sentiment Analysis (ABSA), which we outline in 1.2.1. Additionally, the "task sampling schemas" we investigate in this paper can be generalised to any subfield of ML where Multitask Learning is useful, so in particular Computer Vision applications would benefit from this work. We then briefly present ideas from Transfer Learning, Multitask Learning and Meta Learning (Sections 1.2.2, 1.2.3 and 1.2.4) and indicate how we intend to apply them to fulfill the aims of this project, which we outline in Section 1.3. All of these ideas are revisited in more detail in the Background, which is our literature review, as this section intends to be an introduction to the project as a whole.

1.1 (A brief) History of NLP

Natural Language Processing (NLP) is a subfield of Machine Learning (ML) that attempts to solve a wide variety of inference or understanding tasks on (human-generated) text data. Traditional NLP existed prior to and outside of ML, and this usually involved crafting task-specific features[1] and developing 'rules-of-thumb" based on intuition or problem/domain specific knowledge [2]. Such rule-based early systems relied on deriving heuristics to solve a problem [3], and this approach is in general not robust to natural language variation [4]. Many rule based systems, including regular expressions for string matching [5] and context free grammars for developing parse trees [6] were outperformed after the statistical revolution in the late 20th century, whereby more sophisticated probabilistic modelling techniques opened a gateway into modelling language.

Around a similar time, the term "NLP pipeline" was popularised; referring to the expected sequence of steps that followed when attempting to solve a "difficult" NLP problem. The pipeline is not a formal or concrete one, and there are different definitions

[1] A useful discussion of the various features used for various tasks can be found in [1]

and interpretations (see Figure 1.1 for an example pipeline), but the general ideology is that solving text/speech problems is multifaceted, and relies on a system capable of solving a series of smaller subtasks (in some sense) to achieve its original task [7]. This idea is central to the project, and will be discussed further in Section 1.2.

With the revitalisation of Deep Learning following the A.I winter [8], the field of NLP (like others) have benefitted from major performance increases amongst a range of tasks, due to the increase in both data and compute. Interestingly, Deep Learning provides a mechanism to actualise the distributional semantics hypothesis [9] popularised by Ferdinand de Saussure that is famously summarised by the John Firth quote: *"You shall know a word by the company it keeps"*. The ability to ingest millions of sentences allows us to form representations in the form of word vectors that capture the semantic nature of words, yielding a powerful tool for the NLP field, since these representations can be applied to further downstream models to solve our required tasks.

Task	Description
Part-of-Speech (PoS) tagging	Determining if a word belongs to a class of words that display a similar behaviour syntactically e.g nouns, verbs, adjectives etc
Parsing (Parse Trees)	Determining the relationship between words in a sentence and building a "parse tree"
Named Entity Recognition (NER)	Identifying the entities in a given piece of text
Semantic Roles	Understanding the meaning of each word in the given context
Coreference (Resolution)	Linking words to entities, and resolving disambiguation

Figure 1.1: Potential NLP "pre-task" pipeline required for a model to solve more complex tasks

1.2 Thesis Focus

In this section, we briefly describe the key concepts required for the understanding of the aims of the project. All of these sections are fully fleshed out in Chapter 2, and include comprehensive relevant literature reviews. For a more detailed description of each section, the reader is referred to Chapter 2.

1.2.1 Aspect Based Sentiment Analysis

Sentiment Analysis is a subtask of NLP specifically focused on identifying and categorising the emotion or sentiment of a given document (which we use as a general term to refer to any piece of text, including a word, a sentence, a paragraph etc). Typically, the categories are *Positive* and *Negative* in the binary classification case, but this is often extended to

include *Neutral*, and sometimes *Conflict* (in the case where both positive and negative sentiment are detected).

Aspect Based Sentiment Analysis (ABSA) granularises this notion further, by performing a sentiment analysis task but for each aspect present in the document (for a fixed list of aspects). An example of this is presented in Figure 1.2.

Target Aspect Based Sentiment Analysis (TABSA) goes one extra step, applying ABSA to a set of target words e.g locations in the text. Thus, ABSA is equivalent to TABSA when there is just one target, namely the object of the sentence.

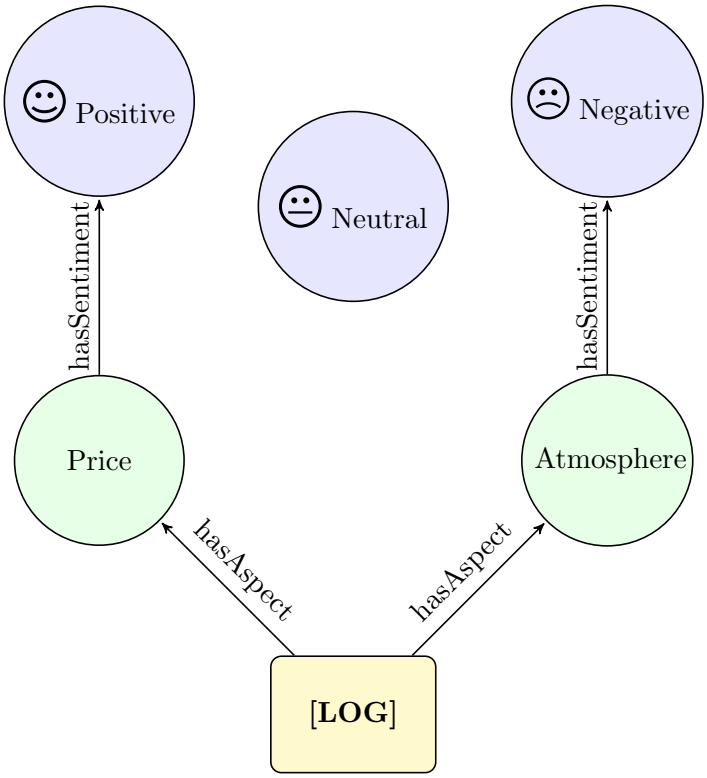


Figure 1.2: Example ABSA Knowledge Graph Construction.
Here, **[LOG]** = "The restaurant was really cheap, but it didnt have a great vibe".
An alternative presentation of the above is that the aspect nodes have sentiment attributes, rather than an additional relation to a seperate node in the graph.

Above, we have used the terminology "log" to refer to an input sequence. This aligns with the companies terminology of referring to a specific response to a survey question as a "log" for that user. See Appendix A for more information.

1.2.2 Transfer Learning

Transfer learning has enabled great progress in the field. The basic premise is that we can pretrain base models on quite general tasks and save the weights, then we can initialise our version of a model using the pretrained, initialised base model and add some additional architecture to *fine tune* the model to our specific task. By fine tuning, we mean randomly initialising some additional architecture that projects onto the task labels for classification tasks or single node for regression tasks and training the model end-to-end on new data. In doing so, we leverage the capabilities of the more general base model by starting in a

(locally) optimal region of the model parameter space, enabling effective transfer learning to the specific task at hand. Typically, the task that the model was used for pretraining was in some sense more general than the task required at fine tuning time, meaning the system has formed some internal representation, i.e. "knowledge", to solve the general task that can then be additionally manipulated to solve the specific task.

Not only do we benefit from the democritisation and distributional nature of research (in that we are able to replicate results without access to the compute or data in which the results were obtained, we can just preload in a set of weights that others have trained) but the notion of transfer learning has foundations in human understanding of our own learning; every time we learn a new task we do not "start from zero", instead we use our pre-existing knowledge as a basis for improvement. Additionally, the type of scenarios that Transfer Learning benefit from are when large datasets are used for the pretraining, enabling us to fine tune onto tasks with significantly less datapoints.

1.2.3 Multitask Learning

To generalise the notion of Transfer Learning one step further, we may believe that a system capable of performing many tasks simulteneously forms a better representation of each task, since it can learn from related tasks during training and adjust its parameters in a way that *shares knowledge across tasks* in the colearning framework.

Suppose we were given a set of tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ that we wanted to learn. Fine tuning/transfer learning would learn each task independently, whereas a clear benefit of multitask learning is that each task is not independent, and colearning encourages a shared knowledge representation. Additionally, this colearning means we can view multitask learning as a form of *inductive transfer*. By introducing an inductive bias, which causes the model to prefer some hypothesis over others, the additional tasks act as a regulariser since the model learns to prefer hypotheses that explain more than one task simultaneously. Another benefit is that suppose we were given a new task τ^* closely related to some subset of tasks $T \subseteq \mathcal{T}$, then we would typically require less data [10] for the new task due to this shared representation of knowledge; from a pedagogical perspective, we often learn tasks first that provide us with the necessary skills to master more complex ones as humans.

1.2.4 Meta Learning

Faced with a new task τ^*, how do we learn it? With a transfer learning approach, we may require tens of thousands of new examples from which we have to fine tune. With a multitask learning approach, depending on the similarity to the tasks trained on, we may need thousands or tens of thousands of additional datapoints. But what if a system could, given a set of tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ for pretraining, learn to generalise well to a new task τ^* with a minimal number of examples. This is the role of meta learning: "learning to learn".

Multitask pretraining to colearn a shared knowledge representation and testing on task τ^* is sometimes referred to as *lifelong learning*, and is a form of meta learning, since the meta learning proposition is to learn "general" properties of a model (e.g weights) that are

highly adaptable to new tasks. Recently, focus has shifted onto model agnostic techniques. In the meta learning paradigm, we explicitly train the model to be able to generalise between the tasks in $\mathcal{T}$ so that, at test time, it is able to learn good representations from a test task from a minimal number of examples.

Like multitask learning, meta learning frameworks act as effective regularisers, since they prefer hypotheses that generalise to many tasks.

1.3 Project Aims & Our Contributions

The primary aim of this project is to investigate the role of multitask learning applied to Language Models for Aspect Based Sentiment Analysis (ABSA) focusing in particular on:

- The role of primary and auxilliary tasks in the training procedure, and identifying suitable secondary subtasks that improve performance in the primary task (ABSA)
- How the training procedure influences performance. We do this by defining some novel task sampling schemas (as per Section 3.2.3) and investigating their effect on the key metrics.
- Investigate the generalisation capability of these systems to previously unseen aspects, including the introduction a novel application of meta learning to language models for generalising to unseen aspects and comparing this to a lifelong multitask learning approach.
- To provide a consistent, open source implementation, of multitask and meta learners applied to language models for the benefit of the research community.

The literature on MTL is extensive, but there has never been an investigation on specific hierarchical subtask structures and sampling schemas in direct relation to performance on downstram tasks. In this investigation, we scope a very general problem and compare it against baseline multitask learning techniques. The motivation for our selected sampling schemas compared to the current multitask learning methodologies is outlined in Section 2.4.2 and defined thoroughly in Sections 3.2.1 and 3.2.3.

In particular, we contribute a detailed investigation of optimal fine tuning procedures in the multitask learning setup to learn complex tasks by relying on supporting subtasks of the complex task, and how best to train an end to end model in terms of sampling from these tasks. We open source our implementation for the benefit of the research community, and this methodology could be applied to any NLP problem and works with a variety of Language Modelling architectures and general downstream classification or regression tasks.

We show that this methodology is valid, in the sense that we outperform all metrics for the (T)ABSA tasks, with our champion model utilising our novel sampling schema. We critically analyse and discuss the results, propose meaningful extensions to this research, and provide useful insights and hypotheses into how the pretraining procedure of large language models can affect the fine tuning procedure. The multitask learning landscape,

in particular the interplay between transformer architectures and how they learn meaningfully, is not yet well understood by the research community; we hope that this thesis provides a platform on which future research can be based upon.

1.4 Commercial Applications

Despite being a research project, this thesis topic has strong connections with commercial interests. This code was integrated into the company codebase and productionised for the automatic generation of knowledge graphs.

The framework created for this project (as referenced in Chapter 3 and Appendix C) was extremely flexible, and so could be implemented for a variety of NLP tasks for either research or commericial purposes.

More information on the company and project relevancy can be found in Appendix A, which provides a lot of motivating material for the decisions in the project, and, although these are noted or cited in the thesis document in the relevant locations, the appendix would be welcome introductory background information for the reader.

1.5 Public Code Repository

The code accompanying this thesis has been made freely available online via GitHub at:

`www.github.com/RyanJenkinson`.

In this repository, you can find the library with all of the code, which we have named `TraMML` - Transformers with Multitask and Meta Learning - as well as this thesis, and all accompanying papers and images. More information regarding specific code implementation and how to run sections of the library can be found in Appendix C or in the `TraMML` README.

Chapter 2

Background

In this chapter, we give a slightly expanded history of the field of NLP, referencing the evolution of language models building up to the current state of the art (SOTA) models used in this thesis. The critical analyses of the literature review will occur primarily on these works, while the preceding sections act as a prelude. Further, we analyse the field of multitask learning applied to language models, with particular focus on the Sentiment Analysis task, and the more granular Aspect Based Sentiment Analysis (ABSA) task that we will be focusing on in the experimentation element of this thesis.

To be consistent throughout this section, we define a *parameter* of a model as a (set of) number(s) that can be learned directly during the training process - e.g a set of weights or biases - whereas a *hyperparameter* cannot be learned directly during training, but has to be defined *a priori* from the user. These hyperparameters will affect the performance of the model, and typically a (grid or random) search is conducted over some reasonable space in order to optimise what these numbers should be.

2.1 The Evolution of Language Modelling

The goal of this section is to outline various methodologies that lead to us modelling language. Our goal is to find some word representation vector that summarises a word in a given context; i.e. representation learning. We ideally want the word vectors of similar words, such as "hotel" and "motel" to be roughly equivalent ($\mathbf{w}_{\text{hotel}} \approx \mathbf{w}_{\text{motel}} \Rightarrow \mathbf{w}_{\text{hotel}} \cdot \mathbf{w}_{\text{motel}} \approx 1$) since they are used in similar contexts to represent similar things. Learning high quality representations can be challenging, since ideally we should model both the complex characteristics of word use (e.g., syntax and semantics), and how these uses vary across linguistic contexts (i.e. to model polysemy).

2.1.1 Problem Setup

To formally define the language modelling setup: our goal is to infer a probability distribution $\mathbb{P}$ that predicts the next word given the *context*. For now, the context will refer to all previous words, but we will explore subsets of previous words and surrounding words as contexts in sections 2.1.2 and 2.1.4. That is, at time t we want to predict word

$\mathbf{w}_{t+1}$ given words $\mathbf{w}_{1:t} = \{\mathbf{w}_1, \ldots, \mathbf{w}_t\}$. Each word comes from a vocabulary V, so that $\forall i : \mathbf{w}_i \in V = \{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(|V|)}\}$.

By Bayes Rule, we may write that

$$\mathbb{P}(\mathbf{w}_{t+1}|\mathbf{w}_{1:t}) = \frac{\mathbb{P}(\mathbf{w}_{1:(t+1)})}{\mathbb{P}(\mathbf{w}_{1:t})} \approx \frac{\text{count}(\mathbf{w}_{1:(t+1)})}{\text{count}(\mathbf{w}_{1:t})} \tag{2.1}$$

where $\text{count}(\mathbf{w}_{1:i})$ refers to the number of instances of the strings containing $\mathbf{w}_1 \wedge \cdots \wedge \mathbf{w}_i$ in some (ideally large) corpus of text. Note the commutitivity of the logical `OR` operation means that sequential structure is not preserved. In addition, this approximation does not scale well with t since the prediction of words given a large context require good counts of very specific long strings in order to get meaningful approximation to the probability, which are unlikely to appear exactly in the corpus. This approach is thus computationally intractable and data inefficient. These can also be referred to as storage and sparsity problems respectively.

2.1.2 $n-$gram models

To deal with data inefficiency, we can make a natural simplification that the *context* is not the entire preceding string, but instead the previous $n-1$ words for some chosen (hyperparameter) n. The concatenation of n words is called an "$n-$gram" [11]. Clearly, the larger the value of n the more context we have, but the more data inefficient the procedure is as we encounter the same problem as before with the specific $n-$gram being unlikely to appear in our text corpus frequently. The lower the value of n means we are disregarding increasing amounts of contextual information, so this represents a tradeoff for researchers. In practice, often 2-grams ("bigrams") and 3-grams ("trigrams") are used.

However, we still encounter the sparsity problem. If our $n-$gram in the numerator does not exist in the corpus then our probability distribution would predict that word having probability 0 of occuring, meaning we are biasing our distribution towards the particular corpus(es) used. A partial solution to this is adding some ε to every word $\mathbf{w} \in V$, called *smoothing*. If the $(n-1)$-gram in the denominator does not exist then our approximate probability distribution is not well defined, but a partial solution is to instead *backoff* to the maximal $(n-k)-$gram that does exist in the corpus. These partial solutions do not fix the overall problem that $n-$gram models tend to exhibit reasonable local structure, but fail to capture long term dependencies and relationships to have any global meaning. We need a smarter model, and one that balances this important interplay between non trivial contextual information and data efficiency.

2.1.3 Pretrained Word Embeddings: Humble Beginnings

In the previous sections, we have described the importance of context to language modelling. The first class of methods that ignited research into word embeddings and representations was the Word2Vec models by Google [12]. They built upon the Skip Gram model [13] to learn distributed vector representations that captured syntactic and semantic word relationships, and proposed extensions that could be applied to the Continuous Bag Of

Words (CBOW) model which they also introduced, as seen in Figure 2.1. In particular, they introduced some key training techniques such as negative sampling and hierarchical softmax which enabled them to ingest orders of magnitude more training data from which greater amounts of contexts could be seen.

Despite the primitive neural network model, the word vectors are specifically trained to predict the surrounding words in the sentence via backpropagation to maximise the average log probability in a context window of size c and thus the vectors can be seen as representing the distribution of context in which the word appears.

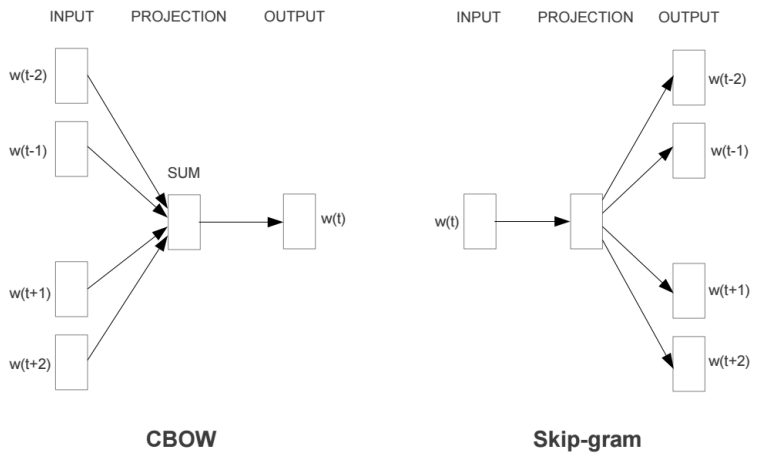


Figure 2.1: Continuous Bag Of Word (CBOW) models vs Skip Gram Models
Skip Gram models predict a words context given the word itself, whereas CBOW models predict a word given its context.

GloVe Vectors [14] take a different approach, and propose a weighted least squares model that looks to factorise the (log) word co-occurance matrix. They show that this approach can be formalised mathematically in the setting of an extension proposed by Mikolov [12], and do additional training tricks such as filtering the data to reduce the weighting factor for more frequent words during training (since these words appear in many contexts, and thus dont carry as much information).

Since the methods they proposed could ingest larger still amounts of context by leveraging age-old scalable mathematical optimsiation routines [14], their results proved state of the art, as well as the fact that they were much more sample efficient, with accuracies around 4% higher than the CBOW or Skip-Gram methods given the same compute time.

The important factor about these methods, is that the word embeddings captured contextual structure. We can look at a lower dimensional representation of the embedding space using dimensionality reduction techniques, for example tSNE [15], which uses a stochastic neighbour embedding that aims to approximate the high dimensional structure in the form of a similarity matrix of conditional probabilities between datapoints with a lower dimensional onevoptimised by minimising the Kullback-Leibler Divergence [16], to find that *similar words occupy similar regions of the embedding space.* This indicates that the embedding space learned captures contextual information, since words close in

the embedding space are "associative neighbours" [2] - this is demonstrated in Figure 2.2. Interestingly, we can define an algebra in this space, in the sense that the relationship between clusters of words are represented by similar vectors (e.g tenses, but also more complex world information such as country-capital).

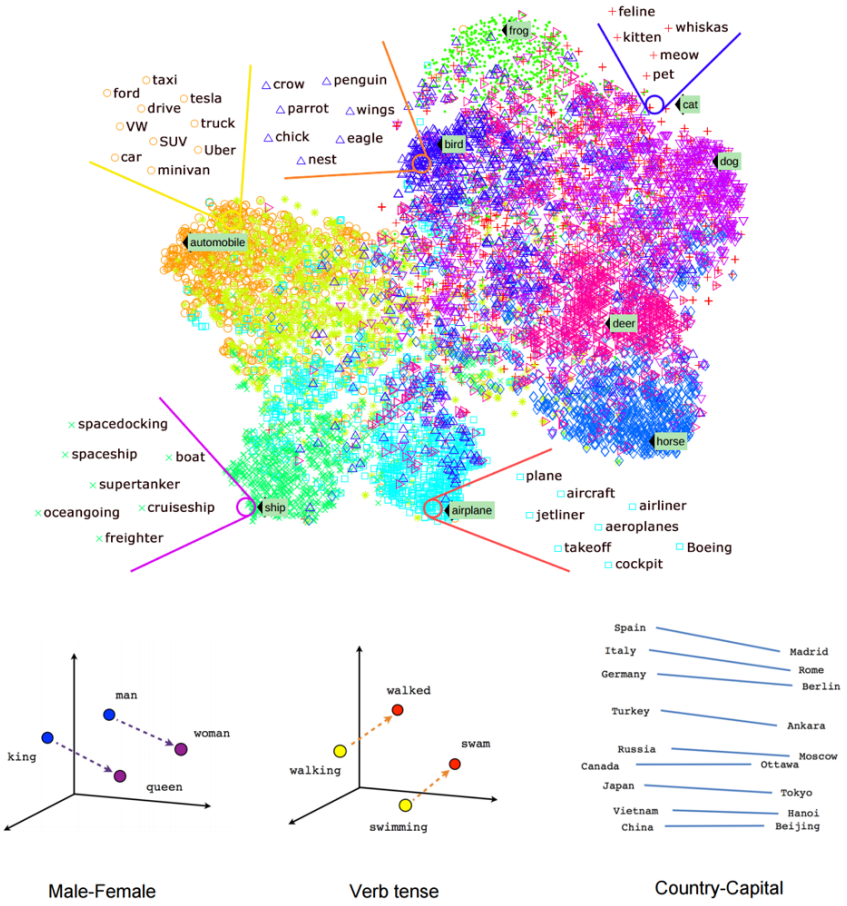


Figure 2.2: The reduced dimensionality word embedding space found using `word2vec` [12] and the algebraic relationship between clusters of words in this space

2.1.4 The Rise of Deep Learning: Using Sequentiality for Context

Deep learning successes tend to come in the large data regime. The distributional semantics hypothesis proposed by Saussure (as discussed in Section 1.1) can apply directly here, since we simply need to ingest lots of data about how words are used and in which contexts and this is, in a sense, all we need to know about the word (e.g we don't need to form any external world view or model, a word is entirely defined by the contexts it is used in). The previous section highlights that word vectors are effective in capturing contextual information, and in this section we explore deeply uni/bi-directional models.

The disadvantage of the methods previously discussed is that it might be good to have a good representation for each word, but how do we combine them? As a baseline we could, once we have a good word representation, just "average the words (embeddings)"

[2] in the linguistic sense, an associative neighbour y of a word x is a word such that when x is replaced by y in a given context, the sentence still makes sense.

to get a sentence embedding. This has its drawbacks, however, in that we lose the word order and sequentiality structure. We focus, now, on networks that take into account the sequential structure that gained traction in the field, that build upon these simply averaging/concatenation models of word embeddings to.

Recurrent Neural Networks

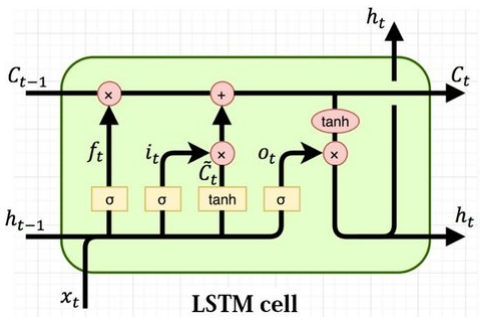
Recurrent Neural Networks [17] are an ideal choice to deal with dynamic length input sequences that are prevalent in all NLP tasks. They are defined by two simple equations: RNNs store a hidden state representation at every timestep (which here corresponds to a word/token in the sentence) and they use this hidden state $\mathbf{h}_t$ to: (a) update its hidden state representation at the next time step whilst including information about the current time steps (i.e word) input and (b) make a prediction based on this hidden state at each time step. They are optimised using backpropagation through time [18]. For input $\mathbf{x}_t$ and hidden state $\mathbf{h}_t$ at time t, weight matrices W_i and biases $\mathbf{b}_{\{h,o\}}$ we can get the prediction at time $t+1$, $\hat{\mathbf{y}}_{t+1}$ with the update rules:

$$\mathbf{h}_{t+1} = \sigma\left(W_1\mathbf{x}_{t+1} + W_2\mathbf{h}_t + \mathbf{b}_h\right) \quad \text{(a)}$$

$$\hat{\mathbf{y}}_{t+1} = \text{softmax}\left(W_3\mathbf{h}_{t+1} + \mathbf{b}_o\right) \quad \text{(b)}$$

However, they suffered from vanishing and exploding gradient problems [19], where the backpropagation of the gradient through the network would prove in practice unstable, since the product of matrices unrolling through time can shrink to 0 or grow to infinity (along some direction $\mathbf{v}$) in value.

The proposed solution to this was introduced was a Long Short Term Memory (LSTM) Network [20], which is similar to a RNN except the inclusion of gates which allow information to propagate through a "cell". It keeps a hidden cell state of accumulated "memory" and decides how much of the information from the next word in the sequence should be incorporated into its hidden state.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 2.3: One LSTM cell with corresponding equations, image taken from [21]. There is one cell for each word in the input sentence. We input embedding x_t, and propagate forward the hidden state h_t and the cell state C_t. $\{i_t, f_t, o_t\}$ are the {input, forget, output} gates, $W^{\{i,f,o\}}$ and $U^{\{i,f,o\}}$ are learnable weight matrices.

Following from this, Graves et al. showcased the power of Deep *Bidirectional* LSTM (DBLSTM) networks for speech recognition temporal classification task [22]. This small yet powerful modification allows the model to account for both sequential directions of

context, and gives a much richer word representation since we can account for downstream words/tokens as opposed to just words/tokens seen prior to the current word at timestep t. It achieves this as per Figure 2.4, by implementing two LSTM models in parallel, one propagating forward through the input tokens and the other propagating backward through the input tokens, and concatenating together the hidden states of each model into a hidden state for the entire architecture. Then, inference is performed as above.

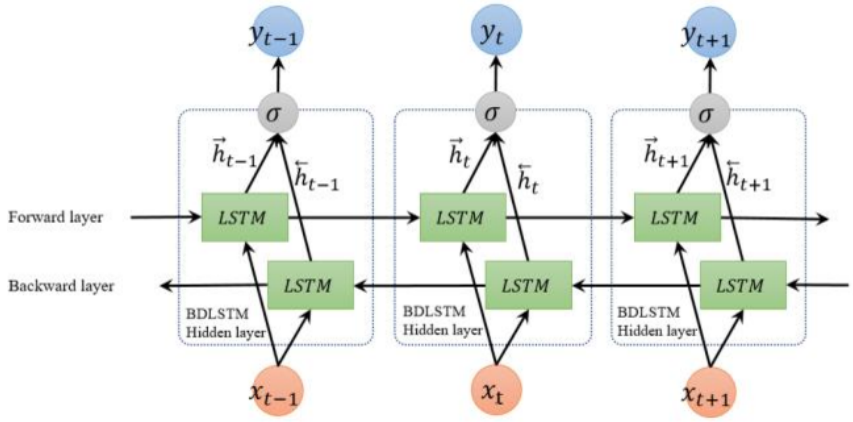


Figure 2.4: Bidirectional LSTM architecture as defined in [22]

ELMo & GPT

There have been many works to get rich context embeddings, but one of the most successful was ELMo [23] (Embeddings from Language Models). Building upon the work of context dependent representations with bidirectional LSTM networks, ELMo embeddings utilise the DBLSTM architecture, but add regularisation and task specific weightings of all the hidden bidirectional layer represenations. Thus, unlike traditional word embeddings in Section 2.1.3, the word representations are functions of the entire input sequence.

ELMo [23] proposes to extract context-sensitive features from a language model. OpenAI's GPT (Generalised Pre Training) Language Model [24], however, enhances the context-sensitive embedding by adjusting the Transformer architecture [25]. Since the transformer architecture is utilised in the current state of the art models, treatment and analysis of this architecture is deferred to section 2.2. While the GPT Model did not outperform SOTA Sentiment Analysis, it achieved SOTA on 9 out of 12 NLP tasks, and provided an important stepping stone to the state of the art language models, showcasing the power of the TransformerXL [26] architecture, which actually enables learning dependency beyond a fixed context width unlike the other recurrent models through having recurrent mechanisms inside the architecture focusing on certain "segments". However, this is still a "standard" language modelling setup in the sense that it is a unidirectional model. The importance of this paper was to set up the following paradigm: *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific downstream task. This is the paradigm we use throughout this thesis, and this paper first demonstrated its efficacy on a range of downstream tasks including absolute improvements of 8.9% on commonsense reasoning tasks, 5.7% on question answering tasks, and 1.5% on textual entailment tasks.

2.2 Current SOTA Language Modelling: Transformers

The previous section provides a basis of understanding on which the current state of the art models are built upon. We outlined important ideas of bidirectional context embeddings, and explore further in this section how state of the art language models have been built. These are the models that were implemented for our experiments (outlined in Chapter 4), and what we built additional functionality on top of.

Following the successes of inductive transfer learning applied to Computer Vision [27, 28], where large convolutional and residual architectures were pretrained on large common image datasets such as ImageNet [29] and COCO [30], focus was redirected onto training large pretrained models that would act as a Language Model. Up until this point, Deep Learning had showed promise in achieving state of the art on many different NLP tasks, but there was rarely a consistent model architecture (since researchers often hand crafted task specific architectures in order to achieve SOTA performance), and these models were often trained from scratch which required large datasets and days to converge. In 2018, researchers proposed Universal Language Model Fine-tuning (ULMFiT) [31], an effective transfer learning method that could be applied to any task in NLP, and introduced key techniques for fine-tuning a language model.

Their aim, as is common to all subsequent language model variants, was to define a model that was expressive enough to, in a sense, "capture language" via a (set of) suitable pretraining task(s). Such a model could ingest large quantities of unlabelled text data, and learn "good" (contextualised) word embeddings. Then, the classifier could be fine tuned (via subsequent linear layers) to capture the idiosyncracies of the downstream target task (especially if the target task was different from the pretraining task). This would require additional data, but the premise of transfer learning is that it requires orders of magnitude less training data than training a model from scratch, since it can leverage pre-existing knowledge in the form of the pretrained word embeddings [32].

2.2.1 Transformer Architecture

While prior models relied on convolutional or recurrent architectures, in 2017 researchers proposed a novel architecture: the Transformer [25], based solely on *attention mechanisms*, dispensing with recurrence and convolutions frameworks entirely. They found in experiments on two machine translation tasks that these models were superior in quality but, more importantly, they demonstrated the models ability to be more parallelizable and requiring significantly less time to train than previous models. This would allow for the ingestion of more data from which to learn in a given time frame and potentially yield much more powerful models.

Attention: An Introduction

Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention had been used successfully in a variety of tasks including reading compre-

hension, abstractive summarization, textual entailment and learning task-independent sentence representations prior to the Transformer network [33, 34, 35, 36], but the Transformer network was the first end-to-end transduction model relying entirely on self-attention to compute representations of its input and output without using any recurrent or convolution elements.

The reader is referred to the diagram in Figure 2.5 taken from the original paper, but we first describe all the components of the diagram in detail.

Encoder and Decoder Architecture

Encoder: The encoder is composed of a stack of N identical layers, with two sub-layers: the first is a multi-head self-attention mechanism, and the second is a fully connected feed forward (linear) layer. The authors utilise a residual connection [27] around each of the sublayers, as well as employing layer normalization [37]. All sub-layers in the model, as well as the embedding layers, produce outputs of dimension d_{model}.

Decoder: The decoder is also composed of a stack of N identical layers: as well as the two sub-layers in each encoder layer, the decoder inserts an additional third sub-layer, which performs multi-head attention over the output of the encoder stack. As before, the authors use residual connections around each of the sub-layers, followed by layer normalization. They also modify the self-attention sublayer in the decoder stack to prevent positions from attending to subsequent positions. They refer to this as *masked self attention* and this, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i (i.e it cannot see itself or into the future).

Attention: The Mathematical Formalism

An attention function can be described as mapping a *query* and a set of *key-value* pairs to an *output*, where the query, keys, values, and output are all vectors of dimensionalities $d_k, d_k, d_v, d_{\text{model}}$, respectively. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

The authors compute the attention function, which they called "Scaled Dot Product Attention" on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V, respectively. Then, the matrix of outputs is computed as:

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

where the inverse of the square root of d_k, the dimensionality of the keys (and also queries), provides a scaling factor since the dot product between the keys and the queries can get large. The first term provides "attention weights" which determine the linear combination of the values V. In essence, we see how similar the query is to each of the keys, then take

a corresponding simililarity weighted linear combination of the values corresponding to those keys.

Multi-headed attention allows the model to jointly attend to information from different representation subspaces at different positions, with the intuition that, as well as providing a form of regularisation to the model, the model can attend to "different types of things in each head" since it forms a specific representation in each one. Formally, we can write:

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

where the projections into each head subspace are parameter matrices:

$$W_i^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}}\times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}}\times d_v} \text{ and } W^O \in \mathbb{R}^{h\cdot d_v \times d_{\text{model}}}$$

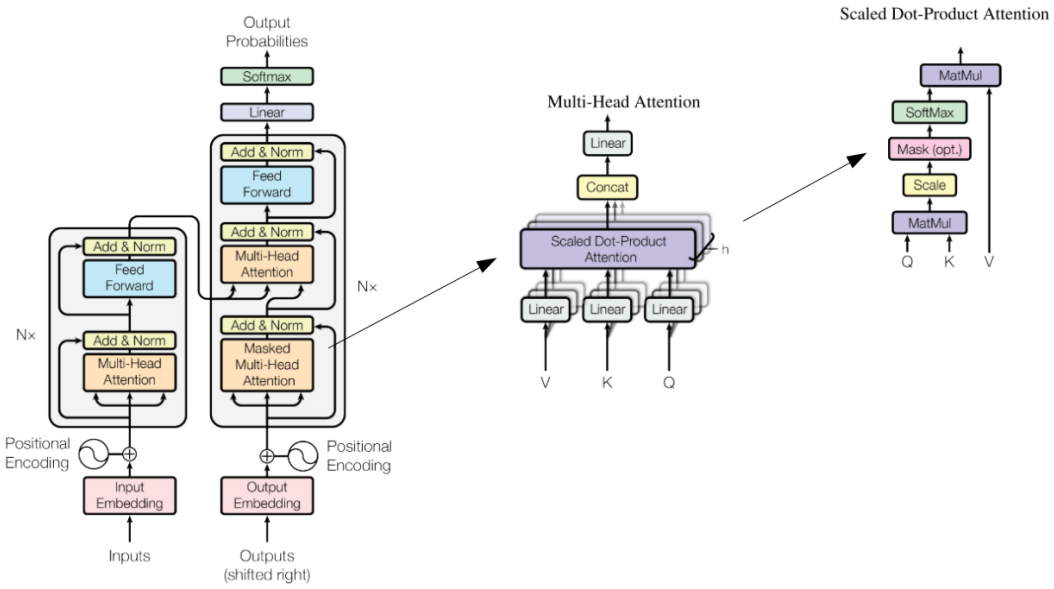


Figure 2.5: The Architecture of the Transformer Model [25]

Benefits of Self Attention

There are many benefits of an end-to-end attention model over traditional recurrent or convolutional models, in particular the computational complexity per layer (compared in detail in Table 1 in [25]), the amount of parallelisation computations (due to the lack of sequentiality needed - a major limitation identified in Section 2.1.4)), and the path lengths the information has to travel in both forward backpropagation passes (of the gradients of the loss through the network) which can destabilise the learning procedure.

The authors note of an additional benefit: interpretability of models. They inspect attention distributions from their models and present and discuss examples in the appendix. They find that individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences, although recent papers [38] have disputed this claim having found that after running ex-

tensive experiments on a variety of NLP tasks, the attention distributions are uncorrelated with gradient-based measures of feature importance, and one can identify very different attention distributions that nonetheless yield equivalent predictions. Research in this area is ongoing, and outside the scope of this thesis.

2.2.2 BERT

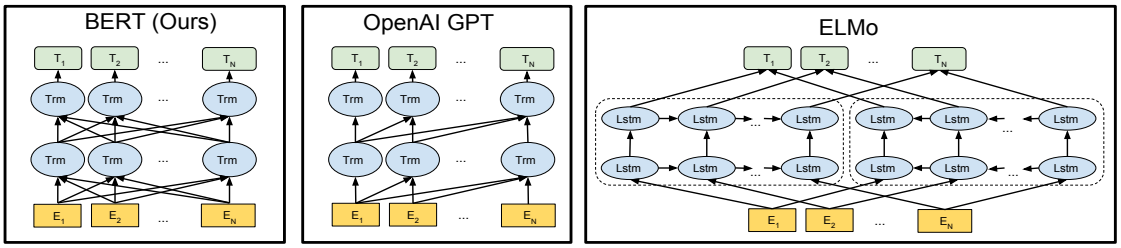


Figure 2.6: BERT vs OpenAI GPT vs ELMo

In 2018, a powerful deep learning model was introduced to the NLP community by researchers at Google. BERT (Bidirectional Encoder Representations from Transformers) [39] was an extremely powerful model, but very different from others around at the time, since it was designed to pretrain deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the learned language manifold was so powerful that fine tuning only required one additional linear layer to be learned for the downstream tasks. Clearly, this was much more sample efficient than previous methods, and it obtained new SOTA performance on 11 NLP tasks, pushing the GLUE (General Language Understanding Evaluation) benchmark [40] up by a 7.6% absolute improvement to 80.4% at the time of paper release. This task is a collection of subtasks that benchmark NLU models on a range of downstream tasks, including Question Answering (QA), Natural Language Inference (NLI) and Sentiment Analysis tasks.

Intuitively, it seems obvious that a deep bidirectional encoding would perform strictly better than a left-to-right or right-to-left model, as well as the concatenation of such models, but traditional conditional language models have not been trained in this way since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context. The authors avoid this by a novel innovation inspired by the Cloze [41] task: the Masked Language Model (MLM). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike traditional unidirectional language model pre-training, the MLM objective allows the representation to fuse the left and right context, which allowed them to pre-train a deep bidirectional Transformer.

Masked Language Modelling - Cloze Task

In the MLM, the idea is to mask out k% of the tokens (the BERT authors consistently use $k = 15$). They augment the token set with a special `[MASK]` token and while this allows them to achieve bidirectionality (since the model does not know which word will get masked, it is forced to keep a contextual word representation of all words in the

sentence) it does come with its downsides: firstly there is a misalignment between the pretraining procedure and test-time procedure, since the `[MASK]` token is never seen at test time. To mitigate this, they do not always replace "masked" words with the actual `[MASK]` token (instead of always replacing the chosen words with `[MASK]`, 80% of the time they replace it with `[MASK]`, 10% of the time they replace it with a random word and 10% of the time they leave the word unchanged in order to bias the representation towards the actual observed word), and secondly, since only k% of tokens (the `[MASK]` tokens) are predicted in each batch, rather than every token for traditional language models, we expect less sample efficiency and more pre-training steps may be required for the model to converge. This indeed turns out to be the case, although recent papers [42] have shown that with simple modifications to the BERT pretraining procedure[3] and a careful analysis of hyperparameters allow the BERT model to perform significantly better, concluding that the original authors significantly undertrained the model and that such a pretraining procedure can match or exceed all post-BERT models in terms of performance on downstream tasks. They called this Robustly optimised BERT approach: RoBERTa.

WordPiece Tokenisation

Another important undermentioned contribution from the paper was the use of WordPiece tokenisation [43] with a 30,000 token vocabulary, denoting split word pieces with the prefix `##`. This allows for common subparts of words to have their own embeddings and thus have a larger general vocabulary since words can be decomposed into their respective word pieces (or `[UNK]` if unknown). For example: `doing` $\rightarrow$ `do ##ing`, since the suffix "ing" is common among many words, and so it has its own embedding. This is particularly useful when subparts of words have their own meanings, e.g in German.

One criticism of the BERT procedure worth noting at this point is that researchers implemented the masking procedure so it applies to *tokens* and not *words*. This was recently corrected, so that when masking a word, if it were split into several tokens then all of those tokens would be masked out simultaneously[4], but the generic BERT-Base and Large models in all languages had this token masking, meaning that words could effectively "see themselves" in terms of embeddings (e.g if a word was split into two, it might be that only one of those tokens was masked out, so in predicting that mask it could use information from the other wordpiece as part of the same word). This isnt exactly the correct data corruption model in terms of predicting whole words, and so the training procedure isnt as effective as it should be.

Next Sentence Prediction Pretraining task

Importantly for this thesis, as we will explore in Section 4.2.1, the BERT pretraining procedure is augmented with a "next sentence prediction" task, with two labels (binary): `isNextSentence` $\in \{0, 1\}$. Many important downstream tasks such as Question Answering

[3](1) training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data.

[4]See the online code implementation update at `https://github.com/google-research/bert`

(QA) and Natural Language Inference (NLI) are based on understanding the relationship between two text sentences, which is not directly captured by language modeling. The authors define segment embeddings to label sentence `A` (the first sentence) and sentence `B` (the second sentence) so that during pretraining they choose sentences `A` and `B` so that 50% of the time `B` is the actual next sentence that follows `A`, and 50% of the time it is a random sentence from the corpus, and backpropagate the loss from this additional task through the model.

Token Structure

The token sequence is prepended with a special `[CLS]` token, which represents the hidden embedding used for classification (with the hopes that the learned contextual representation of this token will absorb contextual information from the sentence(s) fed in). A special separator token `[SEP]` is appended to each sentence (depending on if there is one or two fed into the model) and the sentence is padded up to the maximum sequence length[5] and then an embedding is learned for each token. Predictions are made on downstream tasks as in Figure 2.7.

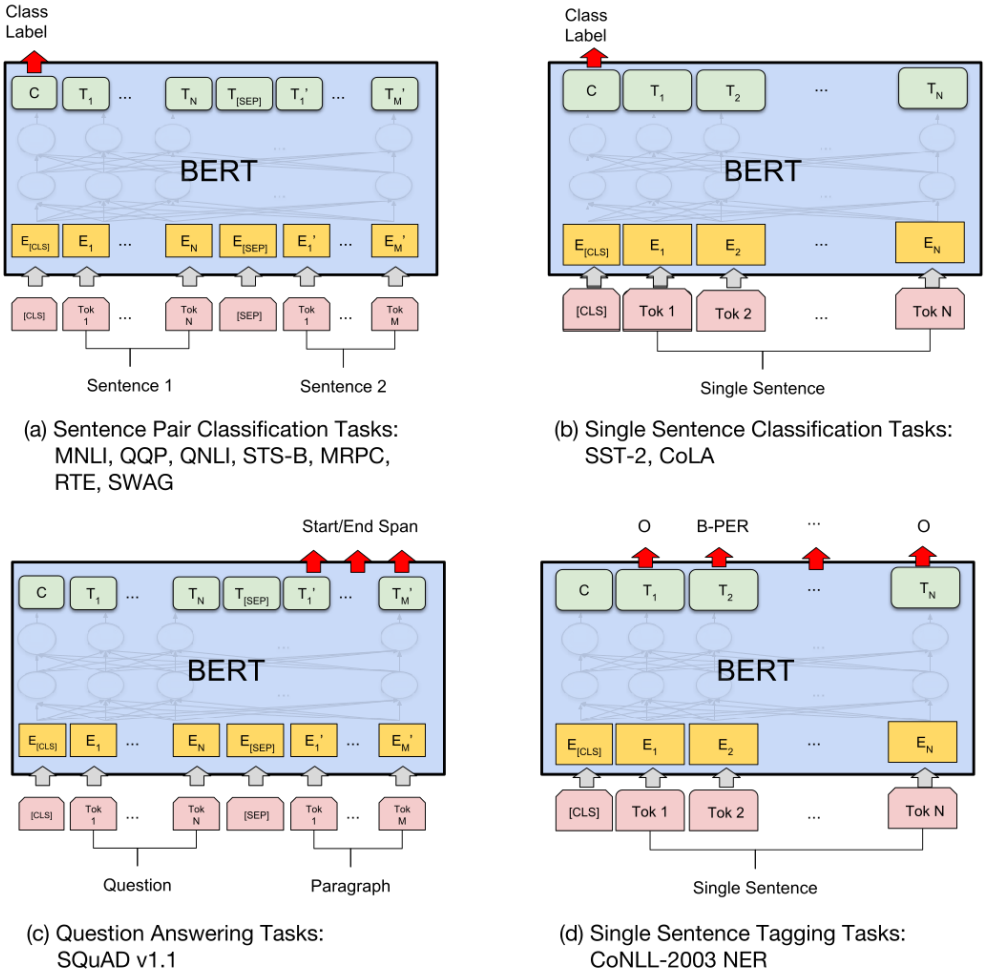


Figure 2.7: BERT Token Prediction

[5] 512 for the initial BERT models

Additionally, the model learns specific segment embeddings, which better enables it to separate out and distringuish the two sentences (if both are fed) as input. Finally, since the locations of the tokens in the input sentence matters, and gives termporal information to the model. This is because Transformers do not encode the sequential nature of their inputs [25] and so this is added as an additional embedding. The three embeddings are summed together, as per the diagram in Figure 2.8.

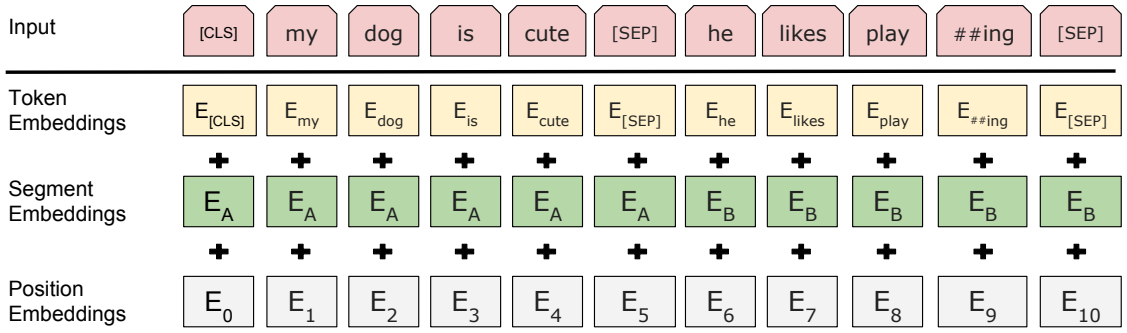


Figure 2.8: BERT Embeddings. In addition to the learned token embeddings, there are segment embeddings (as to whether or not we are in the first sentence or second sentence) and positional embeddings [25] that help indicate the relative positioning of tokens in the sentence.

Ablation Study

The original BERT ablation study looked at 3 model variants:

1. A model with a MLM but without the Next Sentence Prediction (NSP) task
2. A unidirectional left-to-right LM without NSP
3. Model above + biLSTM (randomly initialised) thereby "strengthening" the above model

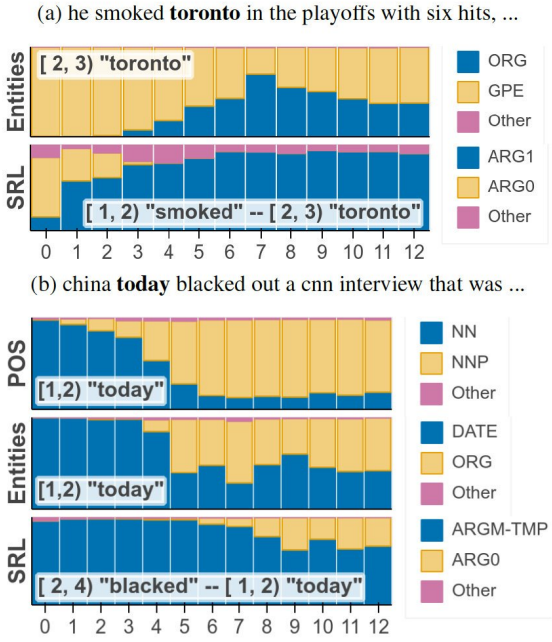
They showed (Table 5 in [39]) that including the NSP task improved the model across all tasks, with major improvements, as expected, on the QNLI task (Question Natural Language Inference, a subtask of the Stanford Question Answering dataset). Additionally, they found that the bidirectional model outperformed the unidirectional one.

While subsequent studies (RoBERTa [42], XLNet [44], "Fair XLNet" [45]) have agreed with the latter claim, they both disagree with the former, instead finding that next sentence prediction as an auxilliary pretraining task *harms model performance*, including on Question Answering and NLI datasets such as SQuAD and MNLI as well as Sentiment tasks (SST-2 - See Section 2.3.1). Additionally, in the "Fair XLNet" study, they report different best performing BERT models, claiming that the BERT model trained without the NSP task does perform better than the one trained jointly with it [45]. This juxtaposition is explored and tested further in our experimental setup in Section 4.2.1.

Recreating the NLP Pipeline

In Section 1.1, we introduced the NLP pipeline, a notional idea that smaller subtasks are required as "building blocks" of language understanding, and that any model capable of

understanding such downstream language tasks must, in some sense, be able to reconstruct this pipeline. Recently, an interesting paper [46] conducted by researchers at Google remarked that certain liguistic information was captured in various layers of the network. By introducing a technique they call "edge probling" they conclude that different layers of the BERT network can resolve syntactic and semantic structure within a sentence, indicating that the model has some explainability and parallels to human cognition in terms of how it understands language. This makes BERT a prime Language Model for our downstream tasks, since it gives us confidence that this model is able to utilise these subtasks for more important tasks of interest.



In the first example, the model originally (incorrectly) assumes that *"Toronto"* refers to the city, tagging it as a GPE. However, after determining that *"Toronto"* is the thing getting *"smoked"* (ARG1), this decision is revised and it is tagged as ORG (i.e. the sports team). In the second example, the model initially tags *"today"* as a common noun, date, and temporal modifier (ARGM-TMP). However, this phrase is ambiguous, and it later reinterprets *"china today"* as a proper noun (i.e. a TV network) and updates its beliefs about the entity type and the semantic role accordingly.

Figure 3: Probing classifier predictions across layers of BERT-base. Blue is the correct label; orange is the incorrect label with highest average score over layers. Bar heights are (normalized) probabilities $P_\tau^{(\ell)}(\texttt{label}|\texttt{s}_1, s_2)$. Only select tasks shown for space.

Figure 2.9: BERT correcting its predictions along the model depth layers, taken from [46]

2.2.3 XLNet

Autoregressive vs Autoencoding Language Modelling: BERT vs XLNet

The formulation described in Section 2.1.1 is referred to as the autoregressive (AR) model. It intends to factorise the likelihood of an input sequence $\mathbf{x} = (w_1, w_2, \ldots, w_n)$ into a forward product, or a backward product:

$$\text{Forward Product:} \quad \mathbb{P}(\mathbf{x}) = \prod_{t=1}^{T} \mathbb{P}(w_t | \{w_{>t}\})$$

$$\text{Backward Product:} \quad \mathbb{P}(\mathbf{x}) = \prod_{t=1}^{T} \mathbb{P}(w_t | \{w_{<t}\})$$

A parametric model (e.g. a neural network) is trained to model each conditional distribution. Since an AR language model is only trained to encode a unidirectional context (either forward or backward), it is not effective at modeling deep bidirectional contexts which we've seen is important for downstream tasks.

As explained in Section 2.2.2, BERT gets around this problem by instead using an autoencoding (AE) model; the pretraining does not perform explicit density estimation but instead aims to reconstruct the original data from corrupted input (the `[MASK]` tokens in BERT). As we've discussed, these artificial `[MASK]` symbols used by BERT during pretraining are absent from real data at finetuning time, resulting in a pretrain-finetune discrepancy. Furthermore, since the predicted tokens are masked in the input, BERT is not able to model the joint probability using the product rule as in AR language modeling. In other words, BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language [26].

Permutation Language Modelling - An alternative pretraining objective

As a consequence, the authors of XLNet propose a generalized autoregressive method that leverages the best of both AR and AE language modeling and while avoiding their limitations, namely:

- Introducing a novel pretraining procedure: fixing either a forward or backward product factorisation but maximise the expected likelihood over *all possible permutations of the factorisation order*. This permutation procedure enables the model to capture bidirectional context since left and right tokens could feasibly appear "ahead" of the current tokens in the unidirectional model.
- As a generalized AR language model, XLNet does not suffer from the pretrain-finetune discrepancy that BERT is subject to. Meanwhile, the autoregressive objective also provides a natural way to use the product rule for factorizing the joint probability of the predicted tokens, eliminating the independence assumption made in BERT.

This latter point is important, and the example given in the paper demonstrates the key difference: consider a concrete example [New, York, is, a, city]. Suppose both BERT and XLNet select the two tokens [New, York] as the prediction targets and maximize $\log p(\text{New York}|\text{is a city})$. Also suppose that XLNet samples the factorization order [is, a, city, New, York]. In this case, BERT and XLNet respectively reduce to the following objectives:

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New}|\text{is a city}) + \log p(\text{York}|\text{is a city})$$
$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New}|\text{is a city}) + \log p(\text{York}|\ \text{New}, \text{is a city})$$

Thus, XLNet is able to capture the dependency between the pair (New, York) which is omitted by BERT, and this leads to a much more effective training signal since XLNet will always be able to capture denser dependency links given this permutation structure and objective.

Ablation Study

To investigate the efficacy of the alternative permutation language modelling pretraining objective, as well as design choices such as (importantly to this thesis) the inclusion of the next sentence prediction task, the authors include a comparison of 6 XLNet model variants compared to the original BERT-Base implementation [39]. For fair comparison, all models are based on a 12-layer architecture with the same model hyper-parameters as BERT-Base and are trained on only Wikipedia and the BooksCorpus data. They find that the various base models of XLNet outperform BERT on a wide range of tasks, 20 NLP tasks in total. **maybe include some more numbers and metrics**

Interestingly, the authors note that the inclusion of the next sentence prediction task discussed by the BERT authors harms the model performance across a wide variety of datasets including SST-2 and, more interestingly, MNLI and SQuAD (since, as discussed in Section 2.2.2, intuitively we would expect NLI/QA downstream tasks to benefit from NLI pretraining task knowledge), but not RACE (a more challenging reading comprehension task than SQuAD). This feels like an important juxtaposition that is not thoroughly investigated by the authors, and a more rigorous analysis of the next sentence prediction task on NLI and QA downstream tasks should be included, to understand better whether this multitask pretraining objective is a useful feature. Nonetheless, the authors conclude that it is unhelpful, and so omit this auxilliary pretraining objective when training the larger XLNet-Large model. The released set of weights for XLNet-Base do not include this next sentence prediction task, and this is important for a key research question that we investigate in Section 4.2.1.

As discussed in Section 2.2.2, a recent paper [42] showcased that improvements and a more thorough hyperparameter study can lead to BERT, and thus the AE modelling approach, equal or surpass that of XLNet in terms of performance on downstream tasks almost unilaterally.

Comparisons with BERT

Upon release of the XLNet paper, the scientific community felt it was unfair to draw direct comparisons between BERT and XLNet, since XLNet used around 10 times more data compared to BERT. Subsequently, the authors followed up the paper with an addendum in the form of a blog post [45] where they ensured that almost every possible hyperparameter was the same for both BERT and XLNet as well as pretraining on exactly the same data.

The study trains 3 separate BERT models, the first being the original one released by the authors, the second being the original model with whole word masking as released by the authors and the third being BERT without the Next Sentence Prediction (NSP) pretraining task since they found the NSP task to harm performance. Despite the Ablation Study in BERT explicitly finding that this task helped in BERT-Base, the XLNet authors report that when training BERT-Large, the best of three variants was often model 2 (10/13 metrics), but occasionally model 3 (3/13 metrics), and never model 1 (0/13 metrics).

They report that even taking the best of these three models, XLNet strictly outperforms BERT across the board on 13 metrics across 11 tasks, albeit by smaller margins

than before when it used 10x more data, thus concluding that this model is strictly better than BERT. Recently, however, RoBERTa [42] ensured the BERT pretraining procedure was much more robust, leading to equal or better performance in XLNet. These transformer architectures, the interplay between pretraining procedures and downstream fine tuning performance and the relationship between larger and smaller models is not yet well understood by the academic community.

2.2.4 Casing in Language Models

Language models use a tokenizer to preprocess the input words. Both XLNet and BERT use a type of sentence piece tokenisation methodology to split up words into subpieces that are fed in as tokens. There are typically two settings for this, either capitalisation is preserved (`cased`) or all the inputs are preprocessed into lower case (`uncased`).

The BERT models release both `cased` and `uncased` versions, with vocabulary sizes of 28,996 and 30522 respectively. The lesser vocabulary size in the cased version, as well as the fact that there are often two sets of the same wordpiece token (one with capitalisation, one without) means that the BERT-`cased` version tends to have worse learned word representations compared to BERT-`uncased`. The XLNet models are `cased` by default. Furthermore, both XLNet and BERT have "base" versions with fewer parameters, and "large" versions with more parameters.

Typically, researchers use `bert-base-uncased` in their research since it has fewer parameters and better learned word representations. Since the XLNet models are so new, not much research has been conducted using them at the time of writing this thesis.

2.3 Sentiment Analysis

Sentiment Analysis is an important downstream task for language modelling. Indeed, one such benchmark dataset for the evaluation of language models is GLUE [40], a collection of 9 natural language understanding tasks, of which the SST-2 (see Section 2.3.1) dataset is a subtask. Often, we fine tune these aforementioned language models on these tasks and, after a hyperparameter search or combination with ensemble models (using multitask learning, as discussed in 2.4.3), achieve higher metrics scores than custom built models trained from scratch.

A more profound demonstration of sentiment's latent utility in text understanding was a paper [47] which found that when given sufficient amounts of capacity, training data, and compute time, the representations learned by generative models training to generate customer reviews included disentangled features corresponding to high-level concepts, specifically they found a single unit which performs sentiment analysis. They termed this the "Unsupervised Sentiment Neuron". These representations, learned in an unsupervised manner, achieved state of the art on the binary subset of the Stanford Sentiment Treebank (SST-2, as discussed in Section 2.3.1).

Sentiment Analysis has often been viewed as a human aspect due to the complex contextual and semantic factors involved, but this makes it an interesting and difficult

problem for NLP models that are interesting in the academic sense, but also applicable in the commercial setting (cf. Appendix A). In this section, we outline the key Sentiment Analysis tasks that we will focus on in this thesis.

2.3.1 SST-2

The Stanford Sentiment Treebank (SST) is a binary (2 class) or fine grained (5 class) single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment [48]. It is a popular task on which to test Sentiment Analysis performance, and is included as part of the GLUE tasks.

The aforementioned "Unsupervised Sentiment Neuron" LSTM model achieved an accuracy of 91.8 on the Binary SST-2 dataset. With the advent of fine tuning language models, we achieved much better accuracy scores than these custom built models. BERT achieved a score of 94.9, an impressive absolute increase of 3%, further justifying the capabilities of a pretrained language model, and XLNet achieved the (current SOTA) accuracy of 96.8%, utilising an ensemble model of the larger parameter version of XLNet. RoBERTa, without any ensemble methodologies, achieved 96.7% - leading to the conclusion that these language models are very similar in terms of sentiment classification (if not undertrained as in the initial version of BERT - discussed in Section 2.2.2).

2.3.2 SemEval 2014 - ABSA

Aspect-Based Sentiment Analysis (ABSA) aims to identify fine grained opinion polarity towards a specific aspect, and is a more challenging variant of sentiment analysis. It is a vital task from many standpoints, since it provides a more nuanced insight into subparts of a sentence on "aspects of interest". In particular this makes it academically challenging, since a model would have to account for this, as well as commercially useful. There are many different variants of this task, as outline in Table 2.1

Variants	Description
Aspect term extraction	Given a set of sentences with pre-identified entities, identify the distinct aspect terms present in the sentence
Aspect term polarity	given a set of aspect terms within a sentence, determine whether the polarity of each aspect term
Aspect Category Detection	given a predefined set of aspect categories (e.g., price, food), identify the aspect categories discussed in a given sentence
Aspect Category Polarity	given a set of pre-identified aspect categories, determine the polarity of each aspect category if it exists in the input sentence

Table 2.1: Subtasks and variants of ABSA. Typically, the polarity categories are {`positive, negative, neutral, conflict`} where conflict denotes both positive and negative sentiment being expressed simultaneously

Task 4 of the SemEval-2014 Workshop focuses on this task, for which there are 4 subtasks (relating to the variants described Table 2.1). We focus specifically on Subtask 4 i.e. given a set of known, fixed aspects $\mathcal{A}$, and a set of fixed sentiment criterion classes $\mathcal{C}$: for each sentence x in the dataset, of which aspects $\mathcal{A}' \subseteq \mathcal{A}$ appear, classify each $a_x \in \mathcal{A}'$ with one sentiment class $c(a_x) \in \mathcal{C}$. We focus on the restaurant dataset[6] which has:

$$\mathcal{A} = \{\texttt{food, service, price, ambience, anecdotes/miscellaneous}\}$$
$$\mathcal{C} = \{\texttt{positive, negative, neutral, conflict, none}\}$$

At the time the challenge was released (2014, prior to all advanced language modelling discussions in Section 2.2), the champion model was using crafted features based on specifically constructed lexicon features from various datasets, Part Of Speech (PoS) and Named Entity Recognition (NER) tagging (discussed in Section 3.2.2) and word and character level n-grams concatenated into a feature vector and put through a SVM model [49]. This model was specifically domain adapted towards the restaurant task, and as such does not represent the scalability and generalisation capability to other tasks of language models such as BERT and XLNet. Nonetheless, this model outperformed all other approaches, achieving an accuracy of 82.93% which is impressive given the task difficulty.

Recently, BERT, as discussed in Section 2.2.2, was applied to this task but did not yield big performance increases over the traditional, custom built models like the one described above, achieving an accuracy of 83.7% (an improvement just shy of 1%) [50]. However, the same paper proposed a novel auxilliary sentence technique whereby either auxilliary questions or pseudosentences were augmented to the input text and joint classification was performed. There were many variants of this (described in Section 2.3.4), but this method pushed the accuracy up to 85.9% - a 3% increase on the traditional custom built models.

2.3.3 Sentihood - TABSA

Target-ABSA (TABSA) generalises this notion one stage further: applying ABSA for each target entity t. An example from the Sentihood dataset is included in Table 2.2

Target	Aspect	Sentiment
LOCATION1	general	Positive
LOCATION1	price	None
LOCATION1	safety	None
LOCATION1	transit-location	None
LOCATION2	general	None
LOCATION2	price	Negative
LOCATION2	safety	None
LOCATION2	transit-location	Positive

Table 2.2: TABSA Example: LOCATION2 is in central London, and thus extremely expensive, whilst LOCATION1 is often considered the coolest area of London.

[6] motivated by tits similarity to the type of data the company works on - see Appendix A

2.3.4 (T)ABSA and LMs: Auxilliary Sentence Construction Method

As eluded to in Sections 2.3.2 and 2.3.3, the current SOTA method developed this year for solving (T)ABSA style problems includes a novel (pseudo)sentence augmentation method, which researchers show strictly outperforms *just* applying the language model to the problem in the multiclass classification setting [50]. In this section, we detail the method, as it is integral to the thesis, and critically analyse the paper.

NLI-**B**	QA-**B**	Label
location - 1 - general - positive	*is the polarity of the aspect general of location - 1 positive?*	1
location - 1 - general - negative	*is the polarity of the aspect general of location - 1 negative?*	0
location - 1 - general - none	*is the polarity of the aspect general of location - 1 none?*	0
location - 1 - price - positive	*is the polarity of the aspect price of location - 1 positive?*	0
location - 1 - price - negative	*is the polarity of the aspect price of location - 1 negative?*	0
location - 1 - price - none	*is the polarity of the aspect price of location - 1 none?*	1
location - 1 - safety - positive	*is the polarity of the aspect safety of location - 1 positive?*	0
location - 1 - safety - negative	*is the polarity of the aspect safety of location - 1 negative?*	0
location - 1 - safety - none	*is the polarity of the aspect safety of location - 1 none?*	1
location - 1 - transit-location - positive	*is the polarity of the aspect transit-location of location - 1 positive?*	0
location - 1 - transit-location - negative	*is the polarity of the aspect transit-location of location - 1 negative?*	0
location - 1 - transit-location - none	*is the polarity of the aspect transit-location of location - 1 none?*	1
location - 2 - general - positive	*is the polarity of the aspect general of location - 2 positive?*	0
location - 2 - general - negative	*is the polarity of the aspect general of location - 2 negative?*	0
location - 2 - general - none	*is the polarity of the aspect general of location - 2 none?*	1
location - 2 - price - positive	*is the polarity of the aspect price of location - 2 positive?*	0
location - 2 - price - negative	*is the polarity of the aspect price of location - 2 negative?*	1
location - 2 - price - none	*is the polarity of the aspect price of location - 2 none?*	0
location - 2 - safety - positive	*is the polarity of the aspect safety of location - 2 positive?*	0
location - 2 - safety - negative	*is the polarity of the aspect safety of location - 2 negative?*	0
location - 2 - safety - none	*is the polarity of the aspect safety of location - 2 none?*	1
location - 2 - transit-location - positive	*is the polarity of the aspect transit-location of location - 2 positive?*	1
location - 2 - transit-location - negative	*is the polarity of the aspect transit-location of location - 2 negative?*	0
location - 2 - transit-location - none	*is the polarity of the aspect transit-location of location - 2 none?*	0

Table 2.3: **B**inary Auxilliary Sentence Construction Method for example in Table 2.2

NLI-**M**	QA-**M**	Label
location - 1 - general	*what do you think of the general of location - 1?*	Positive
location - 1 - price	*what do you think of the price of location - 1?*	None
location - 1 - safety	*what do you think of the safety of location - 1?*	None
location - 1 - transit - location	*what do you think of the transit - location of location - 1?*	None
location - 2 - general	*what do you think of the general of location - 2?*	None
location - 2 - price	*what do you think of the price of location - 2?*	Negative
location - 2 - safety	*what do you think of the safety of location - 2?*	None
location - 2 - transit - location	*what do you think of the transit - location of location - 2?*	Positive

Table 2.4: **M**ultickass Auxilliary Sentence Construction Method for example in Table 2.2

The authors suggest the following sentence augmentation procedures outlined in Tables 2.3 and 2.4 that augment the input sentence with either a question or a pseudosentence, and have the classification be either binary or multiclass. They instantiate the pretrained BERT [39] model, and fine tune on this modified dataset to show that this procedure

strictly outperforms the base case of just performing multiclass classification on the input sentences alone, conjecturing that the performance improvements from this technique leverage the models ability to perform well at NLI (and QA) tasks due to the next sentence prediction pretraining objective (discussed in Section 2.2.2). The 4 different models achieve different top metrics on the Sentihood task, but they find on the SemEval task 4, subtask 4 (as described in Section 2.3.2) that the QA-**B** model achieves the highest accuracies on the SemEval dataset, as well as the highest F_1 score for aspects and AUC score for sentiments on the Sentihood data. We will refer to this as the TABSA "explosion" method, as detailed in the next section.

2.3.5 Analysis of the TABSA Explosion Method

Since this method converts the (target and) aspect information into an auxilliary sentence, this is equivalent to exponentially expanding the corpus. We refer to this process as "exploding the dataset". For a given unique input text, if there are n_t target entities and n_a aspects, then the datapoint is exploded into $n_t \cdot n_a$ new datapoints, where the secondary sentence that is augmented to the model determined uniqueness of the sentence pairs. On the one hand, this creates a much *sparser* dataset, since many of the labels are of the null class (either 'None' or 0 in the multiclass and binary cases respectively). This would almost certainly harm training if the model were learning from scratch. However, since we are initialising the language model with pretrained weights, the authors conjecture that, since BERT in particular has been trained with a next sentence prediction pretraining objective simultaneously to its Cloze task, the model is able to learn from this data since it already has some prior knowledge about NLI and QA tasks from which it can transfer learn.

The binary explosion is also interesting in the setting where the aspects are nonunique, i.e. in multilabelled data. If the aspects were instead just categories themselves, such as emotions[7] then it could be that a person is feeling multiple emotions simultaneously. This binary approach, since the final layer is a sigmoid, will give us the probabilities of each emotion. Then, multilabel categorisation can be performed if this probability is over a certain threshold (say 0.5).

However, one drawback of this method is the computational considerations when expanding the dataset in this way. The multiclass problem means the instantiation of more parameters for each class head, but the binary explosion means much more forward passes through the model and thus a longer training time overall so there is a tradeoff.

2.4 Multitask learning

In Machine Learning, we often care about optimising a particular metric of interest. In order to do this, we generally train a single model or possibly an ensemble of models that all individually try to optimise this metric to perform our desired task. We then finetune and tweak these models until their performance no longer increases. However,

[7] This is an example we will return to when looking at the industry specific dataset, cf Section 3.2.2

this might be a myopic framing of the problem; while we can generally achieve acceptable performance this way we are ignoring information that might help us do even better on the metric we care about. Specifically, this information comes from the training signals of related tasks. By sharing representations between related tasks, we can enable our model to generalise better on our original task. This is precisely the multitask learning paradigm.

One form of Multitask learning is so called *continual learning* or *sequential transfer learning* [51, 32] which aims to train the model with several tasks in sequence so that it remembers the previously learned tasks when learning the new ones. This method is inspired by the human learning process, since we are capable of continuously accumulating the information acquired by study or experience to efficiently develop new skills. With continual learning, the model should be able to performs well on new tasks due to the knowledge acquired during previous training. We could define transfer learning as a means to extract knowledge from a source setting and apply it to a different target setting, as demonstrated in Figure 2.10.

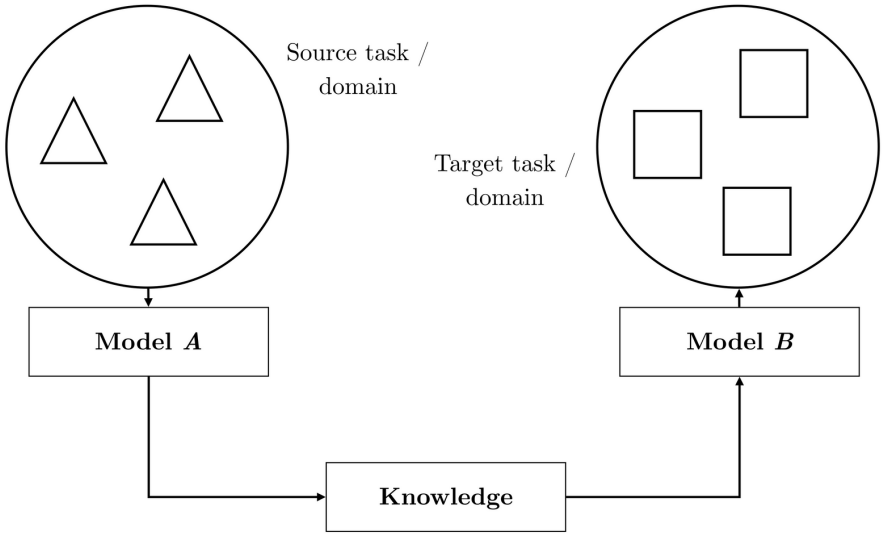


Figure 2.10: Transfer Learning Illustration

Instead of training sequentially, we could train simultaneously (or "jointly"). Our hope is that during this joint training and optimisation of many different loss signals we will get improved generalization on our main task by leveraging the domain-specific information contained in the training signals of related tasks. We could motivate multitask learning biologically: from a pedagogical perspective, we often learn tasks first that provide us with the necessary skills to master more complex techniques. We could also motivate it in a statistical and mathematical context by viewing multitask learning as a form of *inductive transfer*. Inductive transfer can help improve a model by introducing an inductive bias, which causes a model to prefer some hypotheses over others. In the case of joint multitask learning, the inductive bias is provided by the auxiliary tasks and thus acts as a form of regularisation since the model learns to prefer hypotheses that explain more than one task.

2.4.1 Types of Multitask Learning

Hard Parameter Sharing

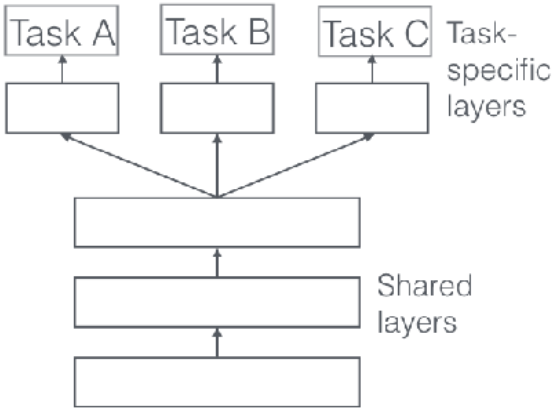


Figure 2.11: Hard Parameter Sharing, with image taken from [52]

One common form of joint multitask learning is hard parameter sharing: we share hidden layers between all tasks, and have task specfic output layers and architectures "on top" of our hidden layers which filter through to the task loss signal. This is an extremely common form for Language Modelling, and what we focus on in this thesis. The motivation is that a learned Language Model, such as those described in Section 2.2, have already in some sense "learned language" and have a reasonable manifold on which the embeddings lie. Then, in order to perform downstream tasks, we just need simple linear layers which act as projection transformations on this space. This is shown diagramatically in Figure 2.11

Hard parameter sharing reduces overfitting. This can be seen mathematically [53] or intuitively: the more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks and the less is our chance of overfitting on our original task.

Soft Parameter Sharing

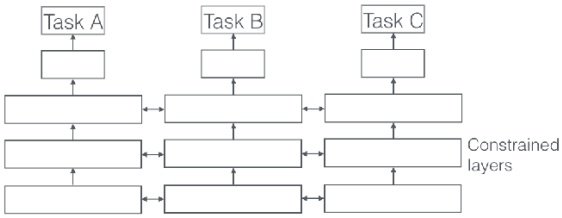


Figure 2.12: Soft Parameter Sharing, with image taken from [52]

Another common method is soft parameter sharing: each task has its own model architecture, but the weights between each layer is "linked" and regularised in order to encourage these parameters to have "similar" values. This is shown in Figure 2.12. It is widely recognised that this method is infeasible for language models due to the large number of parameters, and thus duplication/scaling linearly with the number of tasks is computationally intractible.

Adaption Modules

A middle ground between the two has been explored, where additional architecture is implemented on top of the pre-existing architecture in an attempt to learn the contributions from each hidden layer and weight them appropriately. The most current SOTA review was Projected Attention Layers (PALs) in a paper titled: BERT and PALs [54]. The authors investigated a wide range of these adaption modules, including skip connections and learned layer weighting and propose an alternative attention based architecture (PALs) which use a low-dimensional multihead attention mechanism, based on the idea that it is important to include layers with inductive biases useful for the input domain.

2.4.2 Sampling Tasks

A simple way to train a model on several tasks is to select a batch of training examples from each task, cycling through them in a fixed order. This is often referred to as "round-robin" sampling, but we will refer to it as "sequential" sampling (cf. our definitions in Section 3.2.3). However, a downside to this strategy is that if each task has a different number of training examples then by the time we have seen every example from a particular task we could have looped through another, smaller tasks dataset many times. This could lead to overfitting on smaller tasks, and undertraining on larger tasks. Potentially we could alleviate this issue by manually tuning regularisation hyper-parameters for each task. Alternatively we could employ methods where we see more examples from tasks with larger associated datasets. Concretely, we select a batch of examples from task τ_i with probability p_{τ_i} at each training step, and set p_{τ_i} proportional to m_{τ_i}, the number of training examples for task τ_i. This is the approach of the multi-task BiLSTM of the authors in the initial GLUE benchmark paper [40], and was used by authors in a Hierarchichal Multitask Approach for learning embeddings from semantic tasks [55] . It has the appealing property of selecting each example with the same probability as combining all the tasks and picking examples uniformly (though we train on batches from each task not single examples). This is also the approach used in BERT and PALs [54].

What researchers found by employing these techniques, is that the final metric score does not change too much depending on the sampling schema in this setting; i.e that the sequential sampling mode is sufficient to get good improvements in the multitask learning setting. The main novel approach in this thesis is to redefine the task sampling schemas with a more biologically motivated model. The previous approach can conflate number of examples with "difficulty of a task", since a task with more examples is sampled more. We find when considering certain datasets that this is not always the case, indeed SST-2 has 67,000 training examples but is considered quite an easy task[8] and so this is almost certainly a downside of this technique. Instead, we want to employ a more reasonable strategy: sampling proportional to a hierarchichal task structure, leveraging useful subtask properties and sampling our "main task" more frequently. The details of this methodology are deferred to Sections 3.2.1 and 3.2.3.

[8] the reader is referred to Table 3.1 for a breakdown of our datasets used in this thesis

2.4.3 Multitask learning applied to language modelling

When looking at the leaderboard for the GLUE tasks [40], the best performing models on each task are actually ensemble models that are jointly trained on all of, or a subset of, the tasks. However, these tasks that the models are jointly fine tuned on are generally somewhat unrelated and this thesis proposes a different viewpoint regarding "supporting subtasks" and optimal schemas thereof. For example, In Table 4 of the XLNet Paper [44], the authors present results of multiple settings, including single-task and multi-task, as well as single models and ensembles. In the multi-task setting, they jointly train an XLNet on the four largest datasets—MNLI, SST-2, QNLI, and QQP—and finetune the network on each of the other datasets in GLUE. Only single-task training is employed for the four large datasets to achieve the results posted. This decision is somewhat arbitrary, as described earlier.

ERNIE 2.0

The authors of ERNIE 2.0 [56] propose a way to integrate multitask learning into the pretraining of deep language models. Current pre-training procedures such as the ones in BERT [39] and XLNet [44] described in Sections 2.2.2 and 2.2.3 usually focus on training the model with several simple tasks to grasp the co-occurrence of words or sentences. However, besides co-occurring, the authors argue that there exists other valuable lexical, syntactic and semantic information in training corpora, such as named entity, semantic closeness and discourse relation that these pretraining protocols do not fully capture. Thus, in order to extract the full lexical, syntactic and semantic information from training corpora, they propose a continual pre-training framework named ERNIE 2.0 which builds and learns incremental pretraining tasks through constant multitask learning. This methodology is outlined in 2.13

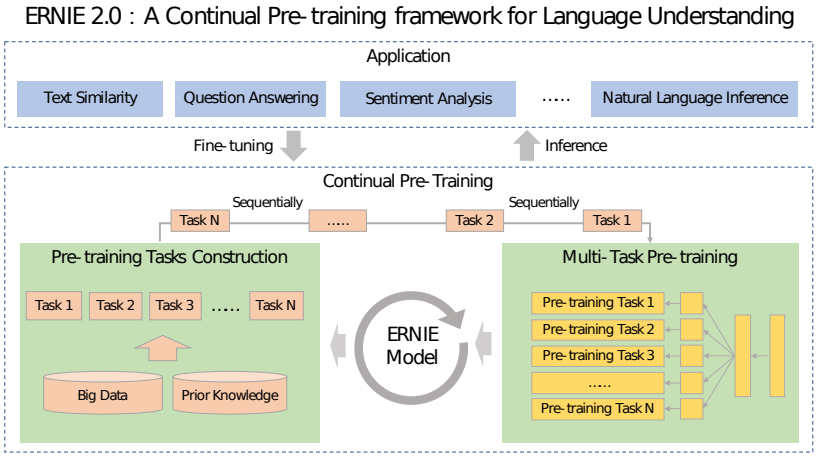


Figure 2.13: ERNIE 2.0 Methodology [56]

While this paper will focus on Multitask learning as a downstream task, the authors of this recent paper flip this on its head, and realise that multitask learning could instead be used to learn better word embeddings at the pretraining step. This novel idea was originally proposed in the extensions to this thesis prior to the paper being released.

The authors split the pretraining tasks into three categories: "word-aware" tasks, "structure-aware" tasks and "semantic-aware" tasks. The word-aware tasks teach the model to capture the lexical information, the structure-aware tasks teach the model to capture the syntactic information of the corpus and the semantic-aware tasks prioritise the semantic signals. These tasks are represented in Figure 2.14, but the reader is referred to the paper [56] since the tasks are not that relevant to describe for the purposes of this thesis, but essentially expand on the dual task pretraining framework of BERT [39].

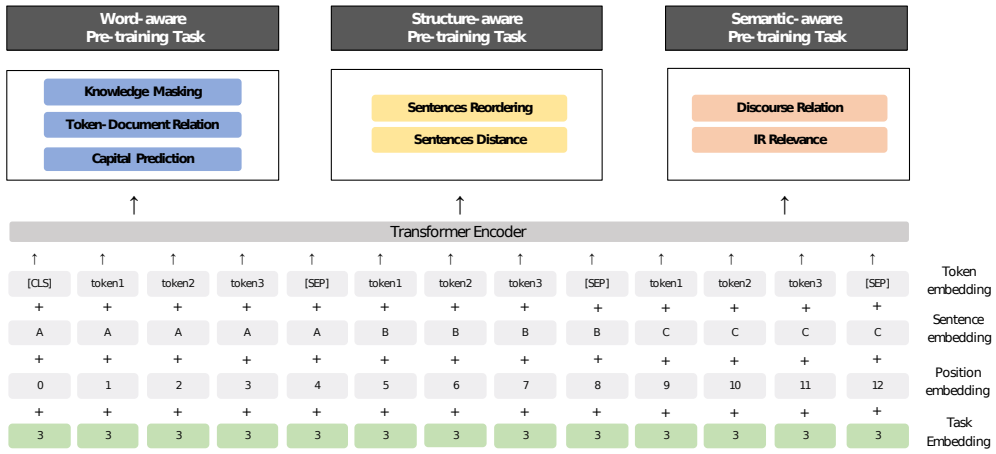


Figure 2.14: ERNIE 2.0 Pretraining Task Framework [56]

2.4.4 Multitask learning applied to ABSA

There hasn't been much work on improving ABSA tasks using multitask learning. We propose to investigate both the classical multitask learning setting of jointly training on tasks, but also generalise this methodology in terms of jointly training via sampling tasks according to a preimposed task importance structure.

2.4.5 Multitask learning for Knowledge Graph Construction

It is clear that from the (T)ABSA methods we can construct the type of Knowledge Graphs (KGs) we motivated at the beginning of the thesis, as per Figure 1.2. Additionally, if combined with NER tasks (or indeed other NLP tasks of interest), this type of framework could also be adapted into multitask multielement learning of the knowledge graph, in terms of the identificiation of entities and then aspect related classification of those entities.

This knowledge graph could then be used for downstream link prediction of missing elements in the graph. An interesting and novel methodology for this, proposed by researchers at Streetbees and in proceedings for the upcoming EurNLP conference [57], proposes converting this knowledge graph into its induced grammar by creating pseudosentences that summarise the original input sentence in terms of key facts (using various weighted walk strategies over subgraphs) and using a language model to end-to-end learn this grammar. Then, link prediction just becomes next token prediction on this vocabulary of entities. However, there are much more standard graph embedding techniques capable of embedding this KG, and optionally performing link prediction [58].

2.5 Meta Learning

Human learning is the benchmark for intelligence, and since humans do not (tend to) learn from thousands or hundreds of thousands of examples, but instead from just a few. We would want our artificial agents to be able to do the same, learning and adapting quickly from only a few examples, and continuing to adapt as more data becomes available. This kind of fast and flexible learning is challenging, since the agent must integrate its prior experience with a small amount of new information, while avoiding overfitting to the new data. In short, we want an algorithm to learn to learn, to be able to learn structured priors from a variety of tasks and be able to apply this prior knowledge effectively on a new task on which it has limited experience of. This is precisely the goal of meta learning.

Suppose we can already solve some set of tasks $\mathcal{T} = \{\tau_1, ..., \tau_n\}$ and we are faced with some new task τ^*. How do we learn it? The transfer learning and multitask learning paradigms state that we could initialise a model with the parameters of some pretrained network, and since this model has learned to solve similar tasks before it might be able to, after a bit of fine tuning, solve our new task. The meta learning framework instead approaches it from a different angle: what if we specifically set up our model training procedure to generalise between tasks? Then, when presented with a new task, since the model has been trying to optimise its generalisation capabilities and learned a set of weights that is "close" to all of the optimal manifolds for each task [59], we still fine tune on our task but its able to generalise to the new task with a limited number of examples since it was explicitly trained to do this, unlike multitask learning.

2.5.1 How we learn to learn

In this thesis we will use a consistent notation for the meta learning terminology, which we outline in this section. Additionally, we will highlight the key changes when going from the traditional learning paradigm to the meta learning one.

In multitask learning, given a set of tasks $\mathcal{T}$, we are training the model to find a set of weights that jointly solves all tasks. In meta learning, given a set of tasks $\mathcal{T}$, we are training our model to learn a set of weights that is able to generalise from any task in $\mathcal{T}$ to any other task in $\mathcal{T}$. This is a more conservative setup and will probably not be able to perform better at each of the individual tasks in $\mathcal{T}$ compared to multitask learning, but it is able to generalise to new tasks much more effectively since that was how the model was trained.

The main difference between meta learning and regular learning is that *datapoints in meta learning are datasets*. In the meta learning paradigm, we aim to improve generalisation onto new datasets (corresponding to new tasks), as opposed to individual datapoints in the regular learning paradigm. This means we shift our training and testing procedure to instead train, validate and test on (small) *datasets*. We have a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. The support set is a set of meta training datapoints, which are themselves datasets of tasks. Task τ_i has associated training and testing datasets $\mathcal{D}^{(i)}_{\text{train}}$ and $\mathcal{D}^{(i)}_{\text{test}}$. The training and testing datasets will be performing K shot learning, i.e learning to clas-

sify from just K examples, and will thus be of size K. The support set and query set will be of size `meta_train_batch_size` and `meta_test_batch_size`, respectively.

2.5.2 MAML

We ideally want an algorithm for meta learning to be model agnostic, and this is what is proposed by Finn et al. in their Model Agnostic Meta Learning (MAML) algorithm [59]. As long as the model is trained by (some variant of) Gradient Descent **ref for gradient descent**, we can use this algorithm to update the weights of our model in a way that is mathematically consistent with the type of behaviour we motivated in the previous paragraph.

In their approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task. In order to get this behaviour, we would want the models internal representation to be broadly suitable for many tasks. Their evaluation shows that this algorithm compares favorably to state-of-the-art one-shot learning methods designed specifically for supervised classification, while using fewer parameters, but that it can also be readily applied to regression and reinforcement learning settings.

Algorithm 1 shows the pseudocode for the MAML algorithm, and one can clearly see that it makes no *a priori* assumptions on the model. There are two key components: the inner training step, which uses some setting of the base model parameters to calculate batchwise updates over tasks and the meta training step, which uses these adjusted model parameters to calculate the loss function, and take a gradient descent step in the average loss direction to update our base model parameters ready for the next inner training loop on a new batch of tasks.

Algorithm 1 Model Agnostic Meta Learning (MAML) [59]

Require: $\mathrm{p}(\mathcal{T})$ - distribution over tasks
Require: α, β - inner and outer learning rates/step sizes
Require: Model f_θ
Randomly Initialise θ, the model weights
while not done **do**
  Sample batch of tasks $\mathcal{B} = \{\tau_i\} \sim p(\mathcal{T})$
  for τ_i in $\mathcal{B}$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$ with respect to K examples ▷ K shot learning
    // Compute adapted parameters with gradient descent (inner training loop)
    $\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$
  // Compute model parameters with gradient descent (outer/meta training loop)
  $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\mathcal{T})} \mathcal{L}_{\tau_i}(f_{\theta_i'})$

2.5.3 Reptile & FOMAML

The Reptile algorithm [60], as described in Algorithm 2, is very syntactically similar to MAML, indeed it is not surprising it is essentially equivalent to the first order version of MAML (named FOMAML) [60] in the sense that they optimise both the expected average gradient of the loss over tasks, bringing the model parameters towards the minimum of the "joint training" problem and the expected average inner product of gradients of different minibatches for a given task, improving generalization, just in different ratios.

Algorithm 2 Reptile [60]

Require: $p(\mathcal{T})$ - distribution over tasks
Require: α, β - inner and outer learning rates/step sizes
Require: Model f_θ
Randomly Initialise θ, the model weights
while not done **do**
    Sample batch of tasks $\mathcal{B} = \{\tau_1, \ldots, \tau_n\} \sim p(\mathcal{T})$
    for τ_i in $\mathcal{B}$ **do**
        Compute $W_i = \text{SGD}_\alpha(\mathcal{L}_{\tau_i}, \theta, K)$ ▷ SGD for K steps on $\mathcal{L}_{\tau_i}$ (K shot learning)
    // Compute model parameters with gradient descent (outer/meta training loop)
    $\theta \leftarrow \theta + \beta \frac{1}{n} \sum_{i=1}^{n} (W_i - \theta)$

2.5.4 Meta Learning applied to Language Modelling

Previous work has looked at an end-to-end meta learning language modelling system which employ LSTM architectures [61], since they mimic the meta learning update rules mathematically. Recent work has attempted to generalise to meta multitask learning by employing a LSTM architecture for each of the tasks [62].

The best application of meta learning in the language modelling space is when trying to generalise between languages, they utilise MAML [59] for low-resource neural machine translation, learning to adapt to low-resource languages based on multilingual high-resource language tasks [63].

We instead propose a model that leverages the pretrained language models described in Section 2.2 in order to fine tune using a meta learning procedure for a set of tasks $\mathcal{T}$ as opposed to multitask learning. This ~~is~~ was a novel application of meta learning to the few shot text classification regime **expand here about why we want to do this**

Unfortunately, on 27th August, a github repo[9] appeared linking to a paper published on Arxiv [64] that essentially does this exact procedure :(- to discuss with supervisors

[9] `https://github.com/zxlzr/FewShotNLP`

Chapter 3

Methodology

In this section, we outline our specific methodology for our experiments and define the problem more fully and technically, in line with the project aims outlined in Section 1.3.

3.1 Hypothesis

We hypothesise that models benefitting from shared related task representations will form better word embeddings for each of the downstream tasks. This is the notion of multitask learning; simultaneously solving many downstream tasks in order to improve performance on each individual task.

We further posit that performance can be improved by looking at *dependent subtasks structures*, and propose to investigate how various task sampling schemas (cf. Definition 3.2.3) of these dependent subtasks improves performance on the main task. In our case, the main task we are focusing on is the (T)ABSA tasks - SemEval and Sentihood - explained in Sections 2.3.2 and 2.3.3 respectively. We utilise the transformation method described in Section 2.3.4 to pass this through a language model. Jointly, we will train related subtasks and see how it affects performance.

3.2 Task Setup

3.2.1 Task Structure

In order to solve the TABSA task, we would require performing an ABSA task on each target term individually, so its safe to assume that these tasks are of near equal importance. In order to solve the ABSA task, we would want a language modelling system to be proficient at the following (sub)tasks:

- Understanding of sentiment → **Sentiment Analysis Task**
- Understanding of aspects → **Part of Speech (PoS)/Named Entity Recognition (NER) Task** from which our model can learn important coreferencing and word disambiguation knowledge from. This is also inspired from the previous SOTA, feature crafted NLP model [49] which used PoS and NER features in an SVM model.

Due to the way we have set up the (T)ABSA tasks, as per Section 2.3.4, one could argue that the model should consider a QA or NLI subtask from which to learn. However, the authors of the auxilliary sentence augmentation technique hypothesise that the reason the BERT pretrained LM is capable of performing so well under this construction is due to the NLI next sentence prediction task during training [50]. This conjecture was largely unsubstantiated or tested in the paper. We thus propose not to include such a task, but a discussion of this is deferred to Section 5.1.2.

We want to formalise this notion of "supporting subtasks" that we use in this thesis, and we do so mathematically in the Section 3.2.3. Then, in Section 3.3, we describe how we use these notions to complete a forward pass of the multitask model. In the next section, we will discuss the datasets we use motivated by the above analysis.

3.2.2 Datasets

As is common in the NLP literature, we refer to "labels" as the target classes, and our models are performing multiclass classification.

Tasks (Datasets)	Properties				Size		
	Priority	# Labels	# Sentiment Classes	# Aspects	Train	Dev	Test
TABSA† (Sentihood)	Primary	2	3	4	2952	872	
ABSA† (SemEval 2014)[10]	Primary	2	5	5	3038	747	
Sentiment Analysis (SST-2)[11]	Secondary	2	2	-	67349	872	
PoS (CoNLL 2003)	Secondary	45	-	-	14041	3250	3453
Sentiment Analysis (IMDB)*	Secondary	2	2	-	19872	9975	19872
Classification† (Streetbees Data)	Primary	2	-	34	24303	3097	

Table 3.1: Datasets used throughout this thesis, including various properties of each dataset. * denotes dataset only used for preliminary testing (cf. Section 4.5.3) † denotes using the QA_**B** method to create the exploded dataset as described in Table 2.3. All sizes are given prior to the explosion of the dataset i.e the raw number of input text sentences for each dataset. In some cases, the dev set is equal to the test set, to match the reporting statistics given in various papers and since we are fixing hyperparameters as per [50] so no hyperparameter tuning is required

Table 3.1 shows the datasets we employ for testing. We decided to focus on the QA_**B** "explosion" method for several reasons:

1. It was SOTA across the board for the SemEval (ABSA) task and achieved the largest F_1 score on aspects and AUC on Sentiment for the Sentihood (TABSA) task

2. The Streetbees data was actually a multi-aspect classification problem, and so we needed to have a probability for each of the aspects so we could map this onto a multi aspect output.

3. The BERT pretraining NLI objective was itself binary - it was next sentence prediction. We hypothesised that this would enable the best performance for BERT

[10] specifically Task 4, Subtask 4, Restaurants dataset

[11] There is an unlabelled test set of ∼1k datapoints, but since reporting is often done on the dev set and is so for the papers to which we are comparing to, we keep the splits despite the heavy skew in training and dev/testing sets

(T)ABSA Data

The reader is deferred to the discussion in Sections 2.3.2, 2.3.3 and 2.3.4 for how this data was prepared, as it is equivalent to the methods utilised in the Auxilliary Construction Method paper [50]. As mentioned, we focused on the QA_**B** method for this thesis.

SST-2

The reader is deferred to Section 2.3.1 for a more in depth discussion of this dataset. We use the standard dataset downloaded from the GLUE benchmark tasks [40].

PoS

For the PoS task, we focus on the CoNLL 2003 Dataset [65] which has become the benchmark for PoS and NER testing. The authors released both an English and German version, on which we focus on the English version. As is common in NLP, we refer to the corresponding "label" which comes from a set of classes, so the goal for this dataset is multiclass classification. The dataset is best illustrated via an example:

Token	PoS Label[12]	Chunking Label	NER Label
U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Table 3.2: An example of the tagging schemas for CoNLL 2003

We choose to focus on the PoS task for our use case. The NER task focuses on entities such as people, locations and organisations. Since our ABSA task requires the model to understand how adjectives are related to nouns, and its only the TABSA task that really requires notions of locations (which are themselves proper nouns) we decided that it was more relevant for the model to work on a PoS task simultaneously as it would generalise better in both the ABSA and TABSA settings.

Streetbees Data

We also take the opportunity in this thesis to apply these methods on corporate data. More information about the company partnered with this thesis, and the type of work they do, can be found in Appendix A. As a brief introduction, the company conducts consumer behaviour surveys. The goal for this particular dataset was to ascertain how individuals felt

[12] A comprehensive list of what these labels mean can be found at `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`. However, the reader should just be aware that these symbols roughly correspond to whether or not the word is a noun, adjective, pronoun, verb etc.

emotionally when buying a product. An ontology was constructed, resulting in numerous categories. This was a multilabelled problem, since it was possible that users could be feeling several emotions simultaneously.

To improve the quality of the labels, we filter the data so that only the labels with at least 100 unique datapoints are included, thus avoiding labels with very sparse support. Figure 3.1 shows a count of unique datapoints with corresponding label categories.

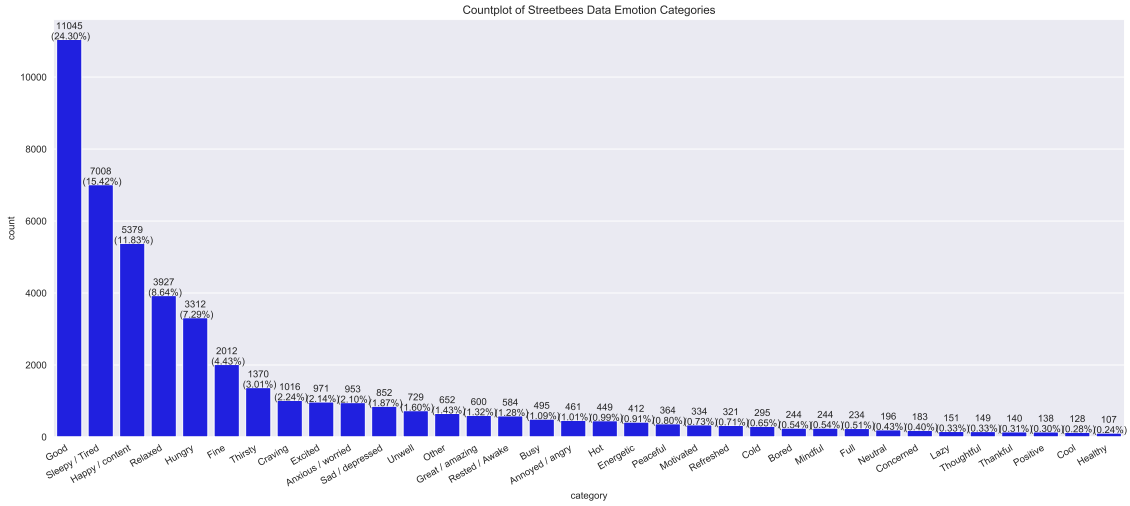


Figure 3.1: Countplot of the Streetbees Data (emotion) categories

We prepare the data in the QA_**B** format as per Table 2.3, with the prepending question string `'Do you feel'` i.e we ask questions `Do you feel` e `?` for emotion $e \in \mathcal{C}$ and have binary labels. Due to the sheer number of classes, we avoid the sparsity problem of the exploded dataset by including all the positive examples as well as 5 negative class samples. This enables the model to learn much better when the number of classes is large, and is in line with the ABSA explosion method in terms of the number of aspects per datapoint.

The core advantage of this binary explosion method is that it gives probability scores of each of the emotion categories, enabling us to select multi labelled output when the probability of an emotion exceeds a certain probability threshold, which we set to the natural value of 0.5.

3.2.3 Task Distributions

We begin by defining some important concepts that will be used throughout. A task is synonymous with a dataset, as per Table 3.1.

Definition 3.2.1 (Sampling Distributions)**.** *A sampling distribution is the probability distribution of the sample statistic. For our purposes, the sample statistic will be a single task drawn from a categorical distribution i.e get* $\tau_i \in \mathcal{T} = \{\tau_1, \dots, \tau_n\}$ *with* $\mathbb{P}(t = \tau_i | \mathbf{p}, e) = p_{\tau_i}$*, where* $\mathbf{p} = \left(p_{\tau_1}, \dots, p_{\tau_n}\right)$ *and* e *is our current epoch (cf. the annealing strategy in Definition 3.2.3).*

In an abuse of terminology, we will refer to the sequential progression through tasks (i.e $\tau_1 \to \tau_2 \to \cdots \to \tau_n \to \tau_1 \to \dots$*) as a sampling distribution, despite its determinism.*

Definition 3.2.2 (Task Weighting)**.** *Define the weight of a task* τ_i *to be* $w_{\tau_i} \in \mathbb{R}$*.*

Definition 3.2.3 (Task Distribution)**.** *The task distribution is our sampling distribution over tasks. With the same notation as in Definition 3.2.1, we in general define* $p_{\tau_i} = \frac{w_{\tau_i}^{\alpha_s(e)}}{\sum_{k=1}^{n} w_{\tau_k}^{\alpha_s(e)}}$ *for strategy* s*. The strategies we are considering are:*

1. ***Random:*** $\alpha = 0 \Rightarrow p_{\tau_i} = \frac{1}{n}$ *i.e uniformily at random sampling tasks from* $\mathcal{T}$*.*
2. ***Prop(ortional):*** $\alpha = 1 \Rightarrow p_{\tau_i} = \frac{w_{\tau_i}}{\sum_{k=1}^{n} w_{\tau_k}}$ *i.e sampling proportional to the task weightings in* $\mathcal{T}$*.*
3. ***Sqrt:*** $\alpha = \frac{1}{2} \Rightarrow p_{\tau_i} = \frac{w_{\tau_i}^{\frac{1}{2}}}{\sum_{k=1}^{n} w_{\tau_k}^{\frac{1}{2}}}$ *i.e sampling proportional to the square root of the task weightings in* $\mathcal{T}$*.*
4. ***Square:*** $\alpha = 2 \Rightarrow p_{\tau_i} = \frac{w_{\tau_i}^{2}}{\sum_{k=1}^{n} w_{\tau_k}^{2}}$ *i.e sampling proportional to the square of the task weightings in* $\mathcal{T}$*.*
5. ***Anneal:*** $\alpha = 1 - c\frac{e}{E}, p_{\tau_i} = \frac{w_{\tau_i}^{\alpha}}{\sum_{k=1}^{n} w_{\tau_k}^{\alpha}}$ *where* c *is a chosen (fixed) annealing constant and* E *is the maximum number of epochs we are training for. Throughout this thesis we fix* $c = 0.9$*.*

Additionally, we consider the ***sequential*** *schedule as a task distribution.*

Remark. The sequential and random task distributions are independent of the defined task weightings. This is intentional, and these sampling strategies will act as a baseline on which we aim to compare results to. Traditional multitask learning is usually done sequentially, often referred to as the "round robin strategy".

A desirable property of the task weightings is that if a task τ_i is *more important* than task τ_j, then we would want a higher task weighting, i.e. $w_{\tau_i} > w_{\tau_j}$. An alternative viewpoint that was initially considered was the "difficulty of the task" in terms of learning ability. One potential proxy for task difficulty is the number of training examples for each task, m_{τ_i} since it could be the case that tasks with less data are harder to learn from. However, we found (cf. Table 3.1) that some tasks that were considered "easy to learn" such as SST-2 have large volumes of data and thus setting $w_{\tau_i} \propto m_{\tau_i}$ seemed unwise.

We instead opted for the following strategy: define each task $\tau \in \mathcal{T}$ to have an "importance" attribute, which is one of {'Primary', 'Secondary', 'Tertiary'}. We then define the task weights as per Table 3.3, where each successively higher tier is weighted twice as much as the previous tier. This is, of course, an arbritrary choice but it should demonstrate the idea effectively. Further discussions around this occur in Section 5.1.3.

Importance of task τ	w_τ
Primary	4
Secondary	2
Tertiary	1

Table 3.3: The task weightings corresponding to each importance level

3.3 Model Architecture Setup

We instantiate a base language model, which is either BERT [39] or XLNet [44], through the `pytorch-transformers`[13] library. We build upon this library rather substantially, however, but this allows us to access the respective embeddings from which we can use on downstream tasks. Specifically, we rewrite a lot of the data processing pipeline code and extracting features code to be cleaner and neater for our purposes, and to deal with the NER task which is not included by default, additionally formalising a standard dataset structure (in the form of a .csv read in by pandas). Then, these features are passed through the base language models as implemented in the `pytorch-transformers` library to get the embeddings.

We build our multitask architecture as follows: for each task $\tau \in \mathcal{T}$, we create a *head*, which is just a simple linear projection of the *sequence summary token*, of hidden dimension d_{model}, to the number of classes for that task, $|\mathcal{C}_\tau|$, which we will refer to as $\mathcal{H}_\tau$. The motivation for this simple linear projection is that we would hope the model to learn a rather complex language manifold, on which subspaces correspond to specific downstream task structures, motivating the linear projection onto this subspace. The sequence summary token is the special classification token that is added by default to each sentence, `[CLS]` for BERT and `<cls>` for XLNet. While the library has functionality to summarise the sentence in numerous ways (such as taking the average embedding of the sentence), we kept it as the default methodology that was enacted at training time of these models.

When running a forward pass of this model, with the aim of jointly multitask learning tasks $\mathcal{T}$, we sample a batch of task τ according to a *task distribution* (See definition 3.2.3) and forward pass this batch through the base language model and the appropriate head $\mathcal{H}_\tau$. Then, the loss is computed and backpropagated through the entire model. The library can support both classification (via Cross Entropy Loss) and regression (via MSE Loss) tasks, but we only consider classification tasks in this thesis. Additionally, there is support in the library for the gradual unfreezing of the base language model layers, but the reader is diverted to Section 4.5.2.

As the number of steps per epoch that the model takes in the multitask setup is not well defined, we have several choices. Since, in this setup, we are sampling tasks randomly, there is no guarantee of going through "every example in every task". Thus, we need a heuristic for how many steps we should take. We want this to collapse onto going through all tasks in the single task setting, and there are many choices. The one adopted for this thesis is the mean of the number of training examples for each task:

$$\texttt{num\_steps\_per\_epoch} = \texttt{mean}(m_{\tau_1}, \ldots, m_{\tau_n})$$

There are many choices for such a function, and it could be the case that a median better accounts for skews in dataset size. However, we stuck with this reasonable heuristic for this thesis. A discussion of this is deferred to Section 5.1.1.

[13] `www.github.com/huggingface/pytorch-transformers`

Algorithm 3 Training Loop Pseudocode

Require: $p(\mathcal{T})$ - distribution over tasks defined by the `sampling_mode`
Require: Optimiser and linear scheduler, initialised with learning rates
Require: Language Model f_θ, and a training batch size `train_bs`
1: Initialise θ, the model weights with pretrained values
2: **for** epoch in `num_epochs` **do**
3: **for** step in `num_steps_per_epoch` **do**
4: Sample task from task distribution $\tau \sim p(\mathcal{T})$
5: Get batch of language model inputs b_i for this task $\mathcal{B}_\tau = \{b_1, \ldots, b_{\texttt{train\_bs}}\}$
6: Pass this batch through the model, computing the loss on this batch $\mathcal{L}(f_\theta(\mathcal{B}_\tau))$
7: Backprop the loss through the model, updating the weights.
8: Update learning rate scheduler and optimiser ready for next step
9: Report metrics at the end of each epoch

3.4 Reporting

We create a folder with an experiment log for the set of tasks $\mathcal{T}$ when running an experiment. This experiment log includes "key statistics" as reported in each of the papers and allows us a simplistic format for the easy comparison between tasks. Additionally, we include tensorboard reporting functionality [66] as well as a `json` metrics report that includes a more granular report of a variety of metrics for reporting, should we need it. The tensorboard logs give us additional validation that the models are training well, and track a variety of metrics for each of the tasks while training. These metrics are outlined and explained in Table 3.4.

Task	Metric	Description	Current SOTA
SST-2	Accuracy	(Total number correct) / (Total number of examples)	0.968
IMDB	Accuracy	(Total number correct) / (Total number of examples)	0.974
PoS	Accuracy	(Total number correct) / (Total number of examples)	-
SemEval_QA_B†	2 Class Accuracy	The accuracy restricted to the two classes {`positive, negative`}	0.956
	3 Class Accuracy	The accuracy restricted to the three classes {`positive, neutral, negative`}	0.899
	4 Class Accuracy	The accuracy of all 4 classes {`positive, neutral, negative, conflict`} - ignoring any aspects with no classification	0.859
	Precision	(True Positives) / (True Positives + False Positives)	0.9315
	Recall	(True Positives) / (True Positives + False Negatives)	0.9083
	F1_Score	(True Positives) / (True Positives + False Negatives)	0.9218
Sentihood_QA_B†	Strict Aspect Accuracy	Calculates the strict accuracy of all 4 aspect categories {`General, Price, Safety, Transit-location`} being correct for a given datum	0.798
	Sentiment AUC	The Area Under the Curve (AUC)‡ for the Sentiment Classes	0.97
	Aspect AUC	The Area Under the Curve (AUC) for the Aspect Classes	0.975
	Precision	(True Positives) / (True Positives + False Positives)	-
	Recall	(True Positives) / (True Positives + False Negatives)	-
	F1_Score	(True Positives) / (True Positives + False Negatives)	0.879

Table 3.4: A table showing all the key metrics we will be reporting as per the literature, as well as brief description of each one and the current SOTA.
† We may refer to these tasks without the prepending "_QA_B" in the thesis with the agreement there is no ambiguity among the tasks described in Section 2.3.4
‡ This metric combines the True Positive Rate (TPR) and the False Positive Rate (FPR) as plotted in the Reciever Operator Curve (ROC) into a singular metric, with 1.0 being perfection and 0.5 being roughly equivalent to random guessing. It is frequent to have ROC scores of greater than 0.95 in SOTA applications.

Chapter 4

Experiments & Results

In this section, we focus on the experiments we ran to validate or disprove our hypothesis: sampling tasks according to an importance subtask hierarchy and utilising multitask learning will improve performance over both multitask learning alone and regular fine tuning/transfer learning. The reader is referred to Section 3.2.3 for an outline of our sampling schemas. We begin with our core experimental results showcasing performance on the ABSA task we are interested in, and supplement this with preliminary and additional experiments that we ran. We wanted to run our experiments over various model architectures, as described in Section 4.1 to additionally test the importance of the Next Sentence Prediction pretraining task, that is currently not well understood by the community, as mentioned in Section 4.2.1. **Write more when done**

4.1 Base Language Model Choices

As eluded to in Section 2.2, we will be utilising the current SOTA language models, namely BERT and XLNet. Due to computational considerations (see Section 4.8.2), we decided to use the base language models with the fewest parameters in each case. This corresponds to models `bert-base-uncased` and `xlnet-base-cased`. The models are described in Sections 2.2.2 and 2.2.3 respectively, and casing is defined in Section 2.2.4.

We hypothesised that casing would be influential to our models, since proper nouns (in the TABSA setting) and all capitals (generally reflecting "anger" which correlates highly with negative sentiment) were present in some of our datapoint examples. However, it was infrequent and in our initial testing we found that the cased model tended to perform worse due to the slightly worse learned word representations on the open source datasets.

Additionally, we wanted to compare fully to the ABSA, BERT and XLNet papers [50, 39, 44], which all used `bert-base-uncased` (for the former two) and `xlnet-base-cased` (for the latter) in their reporting. In order to demonstrate the value of this methodology, we wanted to introduce as few variants from the literature as possible.

Thus, we just consider the `bert-base-uncased` and `xlnet-based-cased` models.

4.2 SemEval (ABSA) Results

In this section we critically analyse our ABSA results, and posit meaningful hypotheses about the nature of our performances in the multitask setting that are pertinent to our main ABSA task. We plot graphs for each metric in a given task, for both models `xlnet-base-cased` and `bert-base-uncased`. We plot a baseline line in grey, which is the performance of the single task model (i.e. regular fine tuning/transfer learning), as well as the SOTA model performance in black. We aim to identify which sampling strategies correlate to good performances, as well as how each of the models perform on the various tasks.

The reader should note that on Figures 4.1, 4.2 and 4.3 the baseline and SOTA line should coincide for `bert-base-uncased` on the ABSA task . However, the original paper had some stochastic elements that could not be exactly recreated, so the model performance is within a very small, reasonable threshold of the paper performance. The SOTA values are as reported in the paper [50].

Figure 4.1 shows the accuracies obtained by each model using all the multitask and sampling mode combinations with the y-axis corresponding to a specific accuracy metric (either 2 class, 3 class or 4 class) for the SemEval task. The reader is invited to look at Table 3.4 for more information on each of these metrics. Figure 4.2 shows the same metrics and plots, but the y-axis instead corresponding to relative increase (or decrease) from the SOTA level. Figure 4.3 shows the precision, recall and F_1 score for the ABSA task, in the same format as the graphs previous.

4.2.1 NLI Pretraining Task Importance

As mentioned in Sections 2.2.2 and 2.2.3, there are key differences in the pretraining procedure of each of our two language models under consideration: the next sentence prediction task. BERT was trained with this binary next sentence prediction task whereas XLNet was not. As discussed in Section 2.3.4, the performance for BERT on this ABSA task using this auxilliary sentence construction method was put down to its ability to leverage knowledge from its pretraining task enabling it to better fine tune at binary QA and NLI tasks. We empirically validate this conjecture through extensive testing that was omitted from the original paper. Indeed, we show this in the results in Figure 4.1. We see that the baseline task performance (i.e just pretraining on SemEval_QA_B) on XLNet was *worse* than BERT on the ABSA task using this method, despite it being a better language model than BERT on 20 "classic" downstream tasks [44]. This suggests that, since the (T)ABSA tasks are more difficult than the NLI/QA tasks in GLUE, it is important to have this NLI pretraining procedure for more difficult downstream tasks. This validates the importance of the NLI pretraining task, and was an important subpart of this investigation. We see, in spite of this, that the XLNet models are able to overcome the SOTA barriers using multitask learning with our task hierarchy. We discuss later, in Section 5.1.2, about how we might have been able to improve performance by additionally incorporating a NLI/QA task at test time, but this was omitted from this investigation.

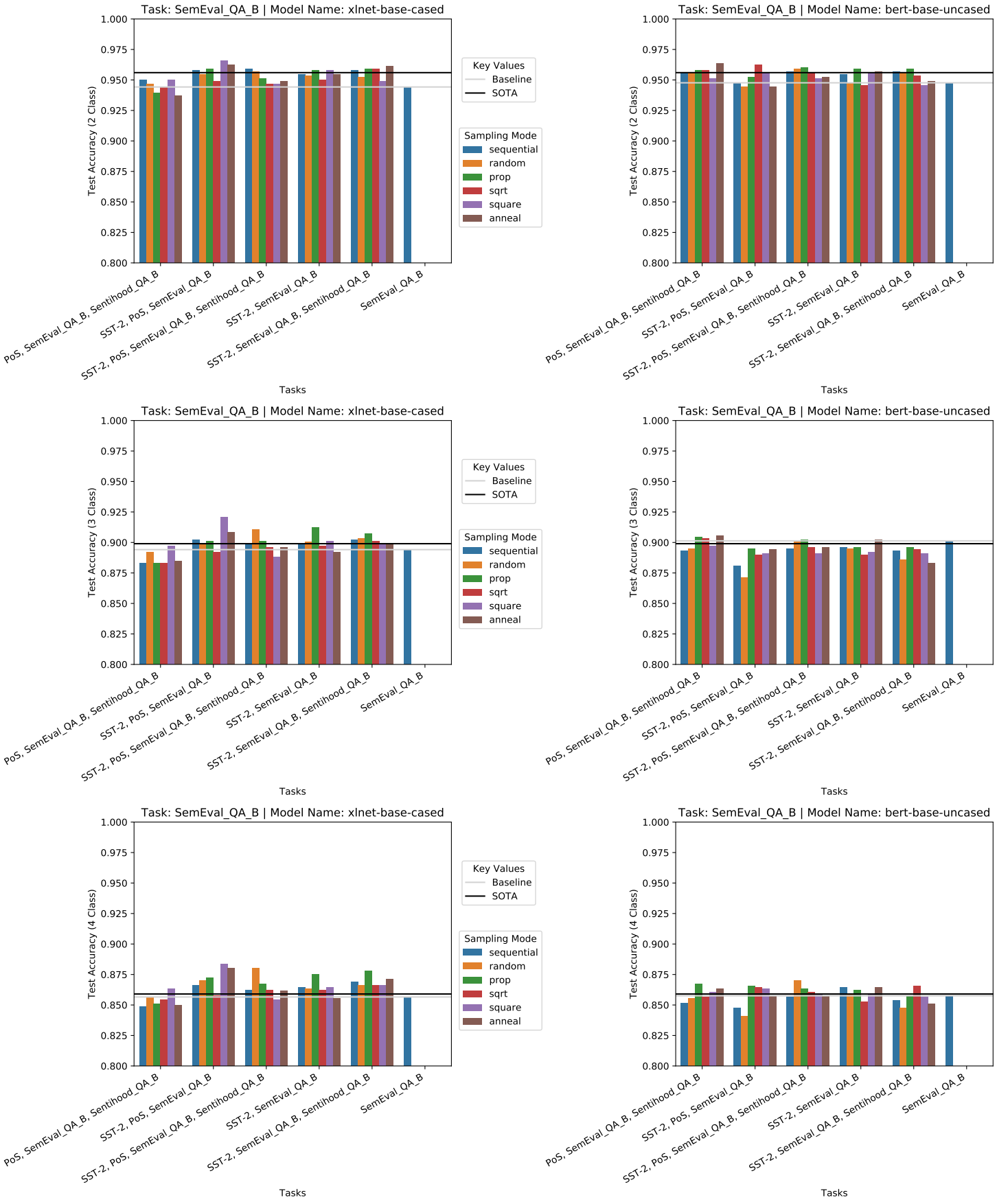


Figure 4.1: SemEval {2,3,4}-way accuracies on BERT and XLNet for our sampling modes

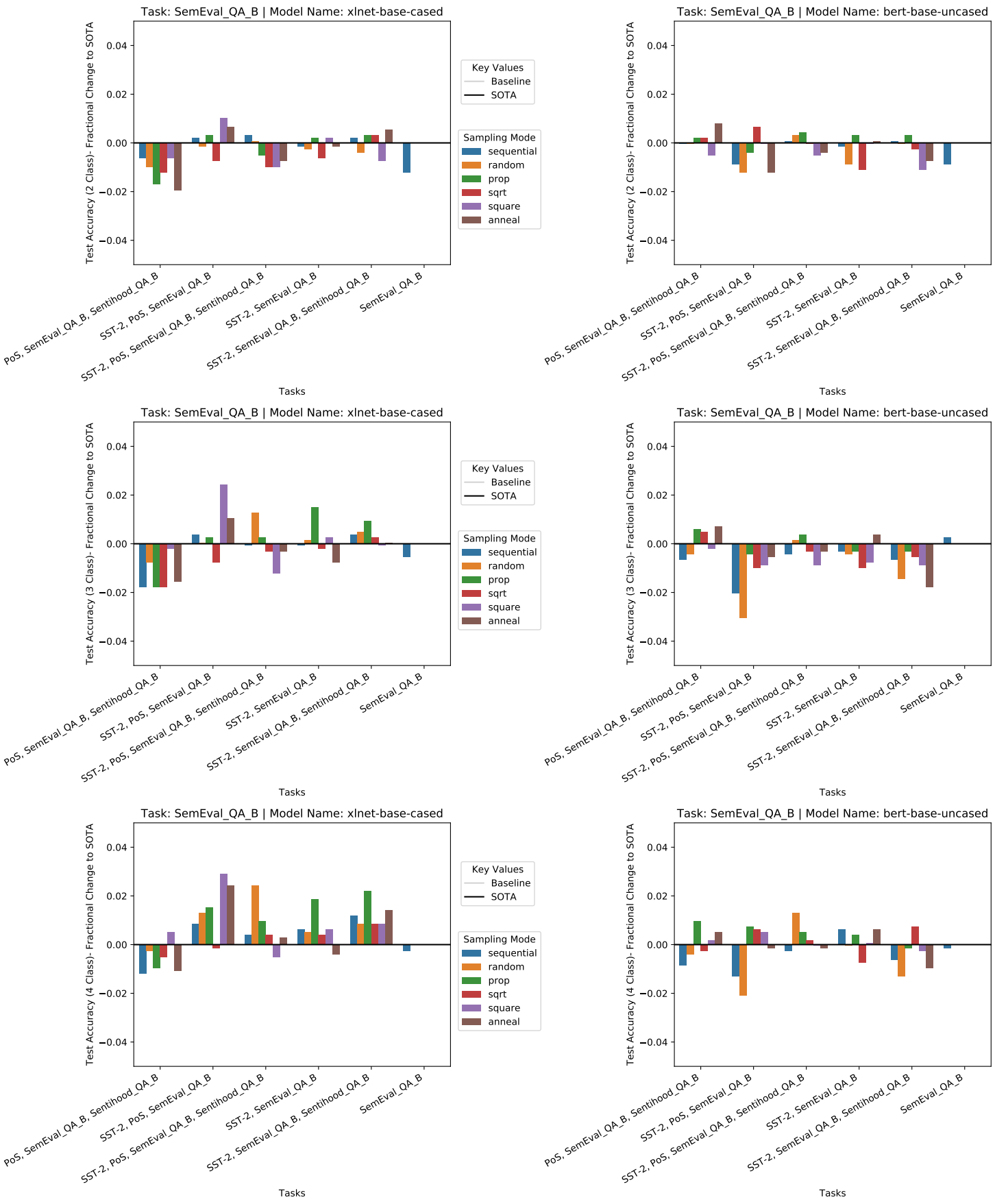


Figure 4.2: SemEval {2,3,4}-way accuracies relative to SOTA performance on BERT and XLNet for our sampling modes

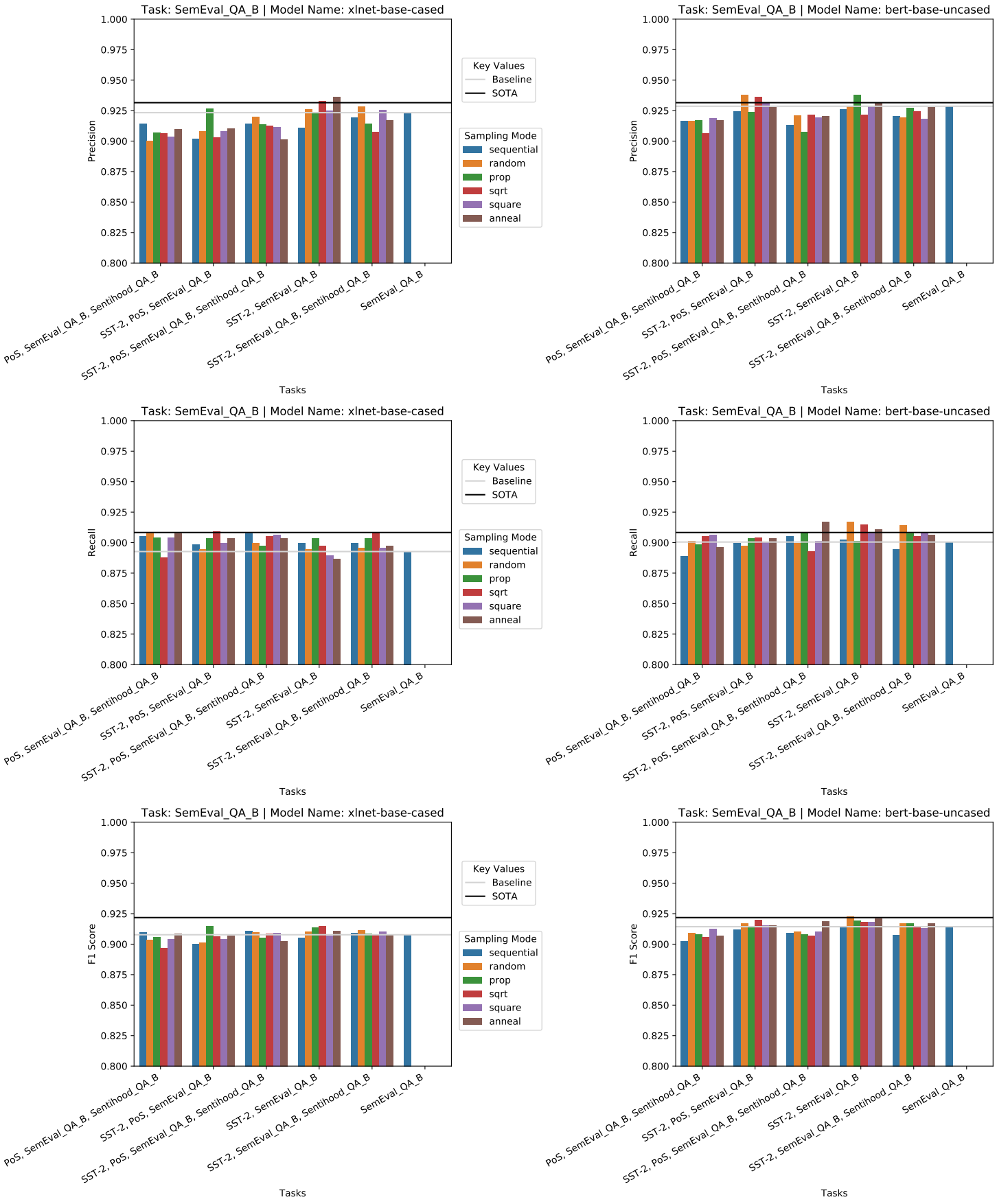


Figure 4.3: SemEval Precision, Recall and F_1 Scores on BERT and XLNet for our sampling modes

While it is true that XLNet is able to recover performance with downstream multitask learning, the benefits of incorporating a QA/NLI task are clear since BERT achieves a better initial performance, making a case for its inclusion as a pretraining task, and thus we conclude that the NLI task must have a role to play in order to better generalise to this complicated ABSA QA_B type setup, as defined in Section 2.3.4. This aligns with the original papers of BERT and ERNIE 2.0 [39, 56], but contrary to XLNet and RoBERTa [44, 42] (although we are using a different, more complicated downstream task than those tested in the papers). This means we can expand and give additional evidence that pretraining with additional tasks does help a models transfer learning abilities in the single task setting.

4.2.2 Analysis of Results

As mentioned in Section 4.2.1, we established that the baseline performance of XLNet is worse than that of BERT, which is interesting in its own right. We see that, when utilising multitask learning, we outperform state of the art according to every accuracy metric as displayed in the tables in Section 4.3. More importantly in terms of experimental validation of our methodology, we see from Table 4.1 that all the models that beat SOTA 2 way accuracy sample according to our preimposed task hierarchy, commonly sampling proportionally rather than sequential or randomly which were our baselines. Unsurprisingly, the top results utilise the multitask representations of the sentiment task as well as the PoS task with the best performing model using the squared sampling schema. This extremises the importance of SemEval_QA_B over the other tasks whilst leveraging representations learned from subtasks of PoS and SST-2 as expected (which makes sense intuitively since both sentiment and PoS labelling tasks are useful in ABSA). This validates the approach and motivation we proposed in Section 3.2.1.

We additionally find that each of the top 5 performing models for binary classification for each of the model classes use the sampling schemas we defined in this thesis as opposed to just sequential sampling, thereby validating the importance of defining these schemas. We report relative improvements in an already high SOTA accuracy score of greater than 1% in the best case. This can be seen in Table 4.1.

Where this multitask learning seems to improve results is in the much harsher metrics. We find that an `xlnet-base-cased` model fine tuning with tasks {SST-2, NER, SemEval_QA_B} using the square sampling schema achieves a 4 way accuracy of 0.8839, which is a 0.02 absolute improvement on this metric and an impressive 2.9% relative increase on SOTA. It seems that these alternative representations enable the model to better classify classes with fewer labels in the dataset (since there is a severe label imbalance for the "neutral" and "conflict" classes), probably since the model does not overfit and learn to optimise just the two label setting, but instead favouring hypotheses that generalise well amongst all tasks (relating back to our discussion in Section 2.4, where the tasks act as regularisers for the model).

We see that, from Figure 4.3, when considering Precision, Recall and F1 Scores, in general we see that `bert-base-uncased` actually performs better than XLNet in terms of

beating the SOTA scores. We also conjecture this is due to the importance of the NLI pretraining task.

Model Name	Tasks	Sampling Mode	Test Acc. (2 Class)	Absolute Increase to SOTA	Relative Increase to SOTA
	NER, SemEval_QA_B, Sentihood_QA_B	anneal	0.963595	0.01	0.79%
	SST-2, NER, SemEval_QA_B	sqrt	0.962457	0.01	0.68%
bert-base-uncased	SST-2, NER, SemEval_QA_B, Sentihood_QA_B	prop	0.960182	0.0	0.44%
	SST-2, SemEval_QA_B	prop	0.959044	0.0	0.32%
	SST-2, SemEval_QA_B, Sentihood_QA_B	prop	0.959044	0.0	0.32%
	SST-2, NER, SemEval_QA_B	square	0.965870	0.01	1.03%
	SST-2, NER, SemEval_QA_B	anneal	0.962457	0.01	0.68%
xlnet-base-cased	SST-2, SemEval_QA_B, Sentihood_QA_B	anneal	0.961320	0.01	0.56%
	SST-2, NER, SemEval_QA_B	prop	0.959044	0.0	0.32%
	SST-2, SemEval_QA_B, Sentihood_QA_B	prop	0.959044	0.0	0.32%

Table 4.1: SemEval_QA_B 2 way (binary) accuracy results (restricted to the top 5 of each model that beat SOTA performance)

4.3 An Overview of SOTA Results Achieved

Metric	SemEval_QA_B			
	Previous SOTA	Our SOTA	Model Name	Tasks (Sampling Mode)
F1_Score	0.9218	0.922926	BERT	SST-2, SemEval_QA_B (random)
Precision	0.9315	0.938071	BERT	SST-2, SemEval_QA_B (prop)
Recall	0.9083	0.917073	BERT	SST-2, SemEval_QA_B (random)
2 Class Accuracy	0.9560	0.96587	XLNet	SST-2, NER, SemEval_QA_B (square)
3 Class Accuracy	0.8990	0.920863	XLNet	SST-2, NER, SemEval_QA_B (square)
4 Class Accuracy	0.8590	0.883902	XLNet	SST-2, NER, SemEval_QA_B (square)

Metric	Sentihood_QA_B			
	Previous SOTA	Our SOTA	Champion Model	Tasks (Sampling Mode)
F1_ Score	0.879	0.928814	XLNet	SST-2, SemEval_QA_B, Sentihood_QA_B (prop)
Aspect AUC	0.975	0.978023	XLNet	SST-2, Sentihood_QA_B (sqrt)
Sentiment AUC	0.970	0.973881	XLNet	SST-2, SemEval_QA_B, Sentihood_QA_B (sequential)
Strict Aspect Accuracy	0.798	0.876201	XLNet	SST-2, Sentihood_QA_B (prop)

Table 4.2: SOTA results for the TABSA tasks: SemEval and Sentihood

We see that on the TABSA and ABSA task, of which there are 10 key metrics, 7 out of 10 (70%) of our champion models utilise our task hierarchy sampling schemas, with the most popular being square and proportional.

We see that, whilst improvements do come from the newer model (XLNet vs BERT), multitask learning further improves the previous SOTA scores, and sampling using our sampling schemas improves it even more. It is a combination of these three aspects that boosts our SOTA scores up to the levels reported in Table 4.2, particularly on Sentihood (TABSA). The graphs for the TABSA dataset are not discussed in this section, and the reader is referred to Appendix B for an overview of additional findings.

4.4 Streetbees Results

At an early stage in the process, we conducted this methodology using the Streetbees datasets. This was useful in testing the end to end scripts for running experiments across

both models (BERT and XLNet) and a few sampling modes (we chose sequential, since it is the baseline, and prop, since it demonstrates our novel idea effectively). We did not use all sampling modes since the company was happy with a "good" solution utilising this technology, and the models were only intended to provide as a proof-of-concept.

4.4.1 Multilabel Emotion Results

The main company dataset we will be considering is the question *"How do you feel?"*, to which there is a fixed emotion set of 34 nontrivial[14] labels. This data is multilabelled, thus requiring a sigmoid for each label. We thus choose to use the ABSA QA_**B** type setup, since we get a probability of each emotion class by default since that is how the data is set up as well as the fact that these auxilliary supplementary questions mimic the way the question was asked to the users originally. The question that is augmented to the responses are *"Do you feel e?"* for each emotion $e \in \mathcal{E}$, the set of all emotion classes.

	Model Name	Tasks	Sampling Mode	Precision	Recall	F_1 Score
Best Performing (F_1 Score)	`bert-base-uncased`	SST-2, NER, SemEval_QA_B, Streetbees_Mood	prop	0.910938	0.917683	0.914298
	`xlnet-base-cased`	SST-2, Streetbees_Mood	prop	0.900042	0.923563	0.911651
Baseline (Fine tuning, single task)	`bert-base-uncased`	Streetbees_Mood	sequential	0.901068	0.918336	0.909620
	`xlnet-base-cased`	Streetbees_Mood	sequential	0.895202	0.926394	0.910531

Table 4.3: Streetbees Results, showing the best performing model by F_1 Score compared to the baseline fine tuning models

The key metric that we focus on is maximising the F_1 Score, which is the harmonic mean of precision and recall. What we see in the results is that the best performing model for both BERT and XLNet utilise the proportional sampling scheme as opposed to "regular" sequential multitask learning, further validating the utility of these sampling schemas. However, the baseline models still perform admirably, with the XLNet model baseline performing very similarly to the best performing one, and the BERT baseline only achieving an F_1 score of 0.05 less.

While the results are important, what is more useful is the impact the model has had on the business. In a presentation to the CEO, we stress tested the model. It was obviously able to match key words corresponding to the emotion categories, but more importantly it was able to do powerful inferencing steps. One standout example was the phrase *"I miss my brother"*, from which the model was able to infer both sadness and loneliness categorisations; demonstrating the power of these contextualised models. Importantly from a business perspective, the marginal gains achieved by this multitask setup might be considered negligible and there is a tradeoff between good metric scores and the additional training time required for the joint multitask models to train.

The model was then additionally applied to a foodstuff ontology. A similar setup was utilised: this time the users were asked *"What are you eating?"* and certain categories were identified as a labelling/coding schema such as *"Chicken"* and *"Wrap"* with multiple labels possible if the person was eating e.g a chicken wrap, and the responses to

[14] a label is defined as nontrivial if there was at least 100 instances of it in the entire dataset of c. 50k datapoints

the questions were fed into the language models with the augmented question: *"Are you eating f?"* for each f in the food categories. This model enjoyed similar successes to the previous example, but investigations as to its efficacy are still ongoing as the company intends to incorporate image data to form a multimodal model since the text data is much noisier in this instance. This multitask multimodal system is currently being developed by researchers at the company; a component of which is the language models we have implemented for the text classification. All of the multitask language modelling frameworks have been implemented into the companies internal codebase and code reviewed.

4.4.2 Generalisation Capabilities

An interesting development from the internal conversations at the company would be how these models would generalise to a new aspect, or emotion in this case. Since the auxilliary question setup is emotion agnostic, we simply added a new emotion category to the labels: "pride" and fed in input sentences to the model. We found that the model was only really able to ascertain exact key word matches, since it still had a learned embedding of "pride" and was in some sense doing a similarity between the input sentence and this emotion word embedding, but it was unable to capture more nuance such as *"My son just won an award"* where more inference would be required and the model has not been fine tuned to do this.

It is relatively easy for the business to label a few examples of datapoints following a new class label, such as pride, but it is unclear if the model would be able to classify them effectively in this multitask setup, since generally thousands of examples are needed (as the model performed best in practice on more common labels, as is expected). We thus turned our attention to meta learning, and explored its utility in this text classification generalisation space. We return to this idea in Section 5.3.

4.5 Preliminary Experiments

4.5.1 Ensuring Baseline Performance

Since our architecture was much more general than typical fine tuning, and was coded from scratch but building upon a lot of various libraries, we had to ensure that in the case of a single task we recreated the performances quoted in our key literature papers. This was the first challenge. We also introduced a more stable and generalisable data ingestion format, that abstracts away a lot of the hard coded implentations for various datasets. We wrote scripts that converted each of our datasets into this standardised format and release these reformatted dataset (split) publicly.

In particular, we focused first on recreating the SST-2 performances reported in the official BERT and XLNet papers [39, 44], since these papers were well written, with a full overview of hyperparameter choices and results, as well as the fact that the data was taken from a source that is well maintained (i.e not outdated mirror links) and is often cited. Once we recreated the performance for this, we moved on to recreating the performance quoted in the TABSA Auxilliary Sentence paper [50].

This proved more challenging, since their code was uncommented and not written for our use case. Thus, it required a lot of code rewriting and refactoring, in particular of the evaluation logic and the data formulation scripts, in order to reimplement the functionality for our purposes, as well as extending it to the more general setting we were considering. Additionally, we commented the code and abstracted various sections for open sourcing and reusability by the research community. The hyperparameters were loosely covered, and the assumption was made that all those hyperparameter settings not reported were left as the defaults in the BERT paper. With these assumptions, we were able to emulate the base performance of that in the paper.

4.5.2 Gradual Unfreezing of Base Language Model Layers

As mentioned in Section 3.3, the library has support for the gradual unfreezing of the base layers. This was going to be one aspect of the multitask schema that was going to be investigated, with the hypothesis that the instantiation of the base weights provides a good initial starting point, and if we froze all layers except the final embedding layer, but as training progressed started to unfreeze deeper layers through time, we would be able to fine tune our embeddings better. We found that this was unhelpful in practice and the results were not much different from just instantiating the base model and letting gradients backpropagate through the entire model, so was omitted after this early testing stage as to not cloud the results with an unnecessary layer of complexity.

4.5.3 Multitask learning for Sentiment Classification

We began by looking at multitask learning for Sentiment Classification. In particular, we took the two most popular sentiment classification datasets: SST-2 and IMDB and tried initial experiments that would:

a. Test the end to end architecture of our model on multiple tasks, and ensure the reporting was correct, as described in Section 3.4.

b. Ascertain whether different tasks contributed to better learned representations and word embeddings for each seperate sentiment downstream task individually.

These initial experiments combined the remarks in Section 4.7 with the task priorities as defined in Table 3.1. The results are shown in Figure 4.4.

From this, we see that multitask learning does marginally improve performance on the both tasks when we use the `bert-base-uncased` model but does not yield any improvements when using the `xlnet-base-cased` model. This could be because the pretraining of XLNet was more effective at capturing word embeddings that are useful for sentiment tasks, and learning different representations in this instance does not yield any improvements. We concluded that it would not be efficient to include both tasks in our general (T)ABSA multitask learning setup and both performed similarly and, as we discuss in Section 4.7, the number of experiments we run scaled exponentially with the number of tasks. The improvements from learning varied sentiment based word embeddings only

enables the successful classification of a handful of additional examples, and thus, since two sentiment tasks seemed redundant, we decided to continue with *only* the SST-2 task as a supporting sentiment task since it was the most common dataset on which sentiment analysis models are benchmarked.

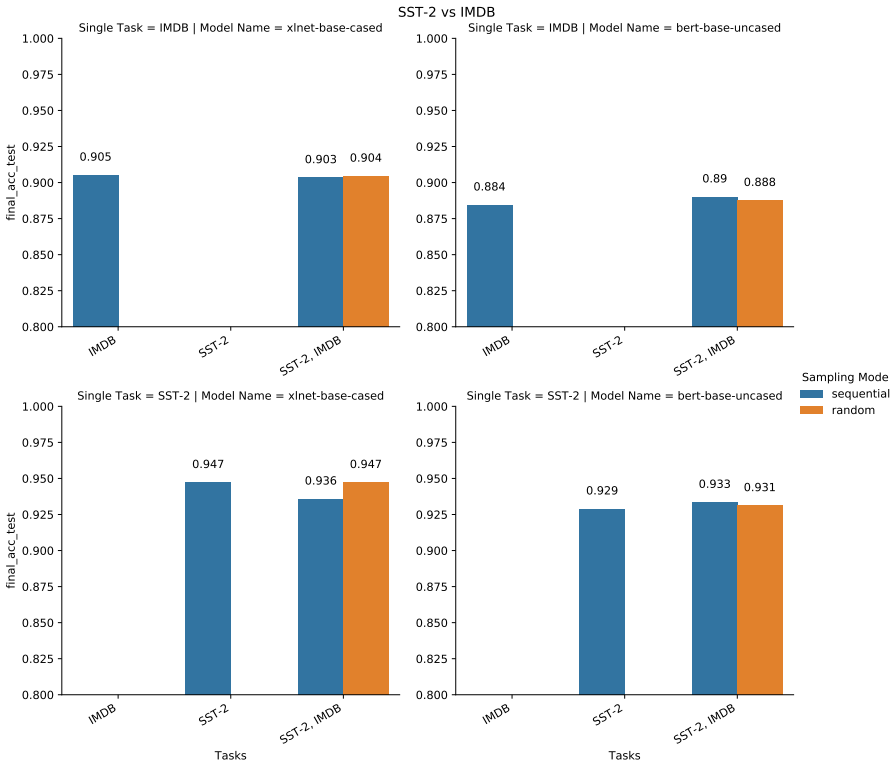


Figure 4.4: SST-2 vs IMDB in the single task and multitask learning setting

4.6 Success of the Sampling Schemas

The previous section highlights the results of the models, showcasing the importance of a sampling schema to best leverage information from related subtasks when attempting to solve a difficult NLP task. We see that, in general, the best performing models do in fact utilise our novel sampling schema in some way, performing better than baseline multitask learning training i.e sequential (or "round robin") task sampling.

Unfortunately, there is no "stand out" sampling schema. In terms of granular accuracies, the square sampling mode clearly wins on the ABSA task, and in general any type of proportional sampling gets SOTA results. The task weightings were arbitrarily defined just to showcase the idea effectively, and so a clear extension is to beter define these task weights. This is one such extension that we discuss in Section 5.1.3.

In general, looking at the Figures 4.1 - 4.3 the sampling schema that often outperforms SOTA is actually the annealing strategy. This strategy is interesting, since it samples proportional to our task hierarchy initially in order to quickly learn good representations, but as the number of training steps increases the sampling schema becomes more uniform over $\mathcal{T}$; a type of smoothing process. This was motivated by the fact that early in the

fine tuning process we want to take gradient descent steps using the batches of our main task and update the weights to good local minima to solve the task, but as time goes on and a good local minima is found, we want to use representations from all tasks to further refine our weights. It seems this strategy works best overall, and investigations that vary the annealing constant might prove a useful extension.

4.7 Remark on Task Distributions for Experimental Optimisation

Although it may look like when we want to run a set of experiments on tasks $\mathcal{T}$ there are:

$$\underbrace{2}_{\#\text{Models}} \cdot \underbrace{6}_{\#Schemas} \cdot \underbrace{\sum_{r=1}^{|\mathcal{T}|} \binom{|\mathcal{T}|}{r}}_{=2^{|\mathcal{T}|}-1 \text{ i.e. } \# \text{ of tasks of length } r=1,2,\ldots,|\mathcal{T}|}$$

such experiments, which scales exponentially badly, there are a few (obvious) tricks we can utilise to reduce the number of experiments, which we break down as a series of observations:

Observation 1: The Single Task Setting

In this setting, all sampling modes are equal, irrespective of the task priority. So, in this setting, we just consider the sequential schema, but really this is just normal fine tuning.

Observation 2: The Two Task Setting

Proposition 4.7.1. *In this setting, if we have two tasks, say τ_i and τ_j of the same task priority, so that $w_{\tau_i} = w_{\tau_j}$, then the task distribution collapses onto two unique settings: the sequential (deterministic) setting, and the random setting.*

Proof. For any $\alpha \in \mathbb{R}$, we have that if $w_{\tau_i} = w_{\tau_j} = w \neq 0$, say, then it follows that:

$$p_{\tau_i} = \frac{w_{\tau_i}^{\alpha}}{w_{\tau_i}^{\alpha} + w_{\tau_j}^{\alpha}} = \frac{w^{\alpha}}{w_{\tau_i}^{\alpha} + w_{\tau_j}^{\alpha}} = \frac{w_{\tau_j}^{\alpha}}{w_{\tau_i}^{\alpha} + w_{\tau_j}^{\alpha}} = p_{\tau_j}$$

and so $p_{\tau_i} = p_{\tau_j} = p$ moreover that (since probabilities sum to 1), $p = \frac{1}{2}$. This can also be explicitly calculated since $p = \frac{w^{\alpha}}{w^{\alpha}+w^{\alpha}} = \frac{w^{\alpha}}{2w^{\alpha}} = \frac{1}{2}$ since $w \neq 0$.

Since the sequential schema is nonrandom, it is different from the random schema, so must also be included. □

The proposition above enables us to eliminate the running of unneccessary experiments and sampling schemas, greatly reducing the computational time required.

4.8 Training Choices

In this section, we detail the specific training choices that were made, including the hyperparameter setup and computing resources.

4.8.1 Hyperparameters

Unless otherwise stated, the experiments were all performed with hyperparameters as defaulted in the ABSA Paper [50], namely those outlined in Table 4.4. Indeed, a thorough hyperparameter optimisation routine would have been far too computationally costly, even just over the most important hyperparameters like training batch size and learning rate. The point of this thesis was to investigate the multitask schema(s) alone, and thus fixing these hyperparameters offers a fair comparison between all schema choices.

Hyperparameter	Default Value
Training Batch Size	24
Learning Rate	2e-5
Warmup Proportion	0.1
Number of Epochs	4
Annealing Constant	0.9
Random Seed	42

Table 4.4: The core hyperparameter default values we used throughout our testing

Additionally, if correlates were found between optimal multitask schemas and performance improvements, hyperparameter tuning could be considered on a much smaller subset of experiments as an additional task. This is discussed in Section 5.1.4.

4.8.2 Computing Resources

All the training was conducted on a single NVIDIA TITAN RTX GPU, named Stanley, which ran 24/7. Training of the experiments (iterating over model architecture choices as per Section 4.1, task distributions as per Section 3.2.3 and tasks as per Section 3.2.2) took circa 5 days, mostly owing to the large number of parameters present in the XLNet architecture.

Chapter 5

Extensions

In this section we propose extensions to our current study, in the form of methodological refinements, which would be quick and easy variants to the investigation we have described, then looking at methodological extensions. We also fully outline our Meta Learning strategy for Few Shot Text Classification, since we were unable to conduct any meaningful experiments due to time constraints.

5.1 Methodological Refinements

5.1.1 Training Time

The number of steps taken per epoch is somewhat of an arbritrary definition, indeed as is the notion of the number of epochs on which to train for. All that matters is the number of gradient descent steps we take, and ensuring that we cycle through our various tasks in a way that reasonably sees "a good quantity" of examples from each task. We chose to do this with a sampling schema and set the number of epochs to match those used in the ABSA paper on which we intended to build upon [50]. As per this paper, the metrics were reported at the end of these 4 epochs, rather than taking the best metrics achieved at some point during the 4 epochs. This means that if we configured an early stopping routine more optimally, we would have almost certainly achieved better results. The reason we chose not to was so that we could compare all of our models to the baselines in the papers, but this methodology has the unfortunate possibility of overfitting in certain cases. In the multitask setting, however, the additional tasks act as effective regularisers; introducing inductive biases into the model require the model to learn representations that satisfy all task hypotheses rather than just the single primary task. In our experimentation and inspection of the loss curves plotted, the models did not drastically overfit, and so this effect is negligible.

There is much research to be done in terms of reasonably understanding when a model should switch between subtasks and at what time in order to gain new knowledge from a different situation. Early Stopping protocols ensure the model does not overfit, but its often the case that base language models are actually underfitting, such as the case in BERT [39].

5.1.2 Subtasks

Of course, the tasks we chose were motivated by the preimposed task hierarchy, but this was somewhat arbritrary - motivated only by human intuition and popular datasets that solve these tasks. We could have additionally added Question Answering (QA) and Natural Language Inference (NLI) tasks, that would almost certainly have improved the performance given the (T)ABSA sentence construction method defined in Section 2.3.4. However, we chose not to do this since we wanted to investigate the importance of the NLI pretraining objective (that was used in BERT, but not XLNet) on performance. Additionally, the implementational details for QA tasks are somewhat more complex, and would have been a timely procedure to implement from scratch and augment our consistent data setup with due to the inclusion of start and end span tokens in particular, but additionally adding these tasks is clearly an extension and left for the interested reader.

Another interesting experimental extension would have been to consider "unrelated" subtasks in this context as a baseline/control case on which to judge experimental results. We felt this wasnt necessary since the single task performance metric provided a reasonable baseline from which we compared all subsequent experimental results, but it would have been interesting nonetheless and would only show to strengthen our hypothesis if the results performed worse than our selected supporting tasks for ABSA. We could hypothesise that generally multitask learning alone helps, and different latent representations produce useful downstream abilities, and the sampling schemas effect is negligible. An interesting hypothesis on which to test is how performance scales with the number of supporting tasks. On the one hand, one would expect diminishing returns as the number of tasks scale, since the optimal manifold on which we want to learn our downstream representations is shared by many of these tasks, introducing a redundency in terms of the input tasks, but it could be the case that we are still slightly underfitting and these additional tasks gradient descent the model into various regions of the parameter space on which better representations can be learned.

5.1.3 Task Distributions

An interesting extensions would be if the task distribution were itself learned in some way, e.g using reinforcement learning techniques. The setup in this paper was somewhat of an arbritrary choice but provides a proof of concept for investigation. Ifthe distribution was itself learnable, one could see if the model learns to prioritise tasks in a similar way to the dsitributions we defined. However, it is possible it would not learn a meaningful distribution and just collapse onto a random structure as this is "good enough" i.e. the model doesn't recieve a strong enough signal from the task distribution as to its relevancy, and there may be an element of hyperparameter tuning involved.

A proposal is a Reinforcement Learning type approach whereby a sampling schedule reflects how the loss decreases for each task - if it decreases the loss on the main task quite quickly then we place more emphasis on it and vice versa. This is a type of Multi Armed Bandit style setup [67], where the action space is deciding which task to select a minibatch from for the next step in training.

5.1.4 Hyperparameter Tuning

As noted in Section 4.8.1, an exhaustive grid or random hyperparameter search would have proved costly given our computational constraints, and was thus omitted from the experiments. Additionally, it is clear that hyperparameter optimisation would improve the performance but, as is common in language modelling research, the space of hyperparameters is far too large to search, and the time complexity of running these operations means only those with access to large scale distributed computing software can reasonably conduct such analyses.

As well as the hyperparameters controlling the architecture of the model, we could also vary the hyperparameters of the optimisers, or indeed the way the model summarises a sentence input (the "sequence summary token" described in Section 3.3) to instead average the token embeddings rather than using the special classification token embedding. All of this functionality exists in the library that was built, but could not be tested over due to the computational constraints this exponentially large search would entail.

5.1.5 Multiple Runs

In our experiments, we fixed a certain random seed in which we conducted all of our experiments. Different random seeds would correspond to different task samplings and stochastic effects in the optimiser. As such, for a more rigorous study, one would want to repeat the experiments several times across several random seeds and take an average. This would make the study much more robust, but was outside the computational constraints of this study.

5.2 Methodological Extensions

5.2.1 Mutual Information

We could utilise a different approach for ABSA that would prove a very interesting extension indeed. We could investigate the role of *mutual information* by introducing a hyperparameter λ that weights the forwards and backwards information from `text_a` (the input sentence) and `text_b` (in the ABSA case, the corresponding pseudosentence/question asked of the input sentence as per Section 2.3.4). Essentially, this is like using the question to predict the response AND using the response to predict the question, and one would expect performance increases to occur if using this methodology.

5.2.2 Label Smoothing

Since neutral sentiment is a very fuzzy subjective sentimental state, training samples which are labelled neutral are unreliable. As such, we should probably employ a label smoothing regularisation term in the loss function, which penalises low entopy output distributions [68]. It can reduce overfitting by preventing the network from assigning the full probability to each example in training by replacing the hard 0/1 targets with smoothed values such

as 0.1 and 0.9. This trick would almost certainly improve performance, since Table 5.1 shows the distribution of sentiment classes in the ABSA Dataset is somewhat disjointed.

Dataset	Positive		Neutral		Negative	
	Train	Test	Train	Test	Train	Test
Restaurants	2164	728	637	196	807	1961

Table 5.1: A breakdown of the sentiment classes in the SemEval ABSA task

5.3 Meta Learning

In terms of the flow of this thesis, we have been able to effectively build systems that leverage alternative task representations to best solve the ABSA task. We described several methodologies to improve performance on this task for use cases in academia or commercially, but we want to extend this one step further.

Given the ABSA task is quite niche in terms of large datasets, and quite granular in terms of the number of examples corresponding to a certain category, and the category classes differ from domain to domain[15], we motivate a use case for the novel application of Meta Learning to Natural Language Processing systems. The outline of the most popular and recent meta learning algorithms can be seen in Section 2.5. Importantly, these algorithms are model agnostic, meaning any function that is differentiable (including our Language Models) can be used in this meta learning paradigm.

We have to set up the problem appropriately for meta learning, and in doing so we must imagine a reasonable task that we are faced with at test time. One such motivation for this work in the corporate setting was that clients often come asking how people felt about a particular category: a hot topic and the source of this example will be the category "sustainability". This category does not exist in the dataset literature, and thus we can only hope to fine tune our model on this category. However, additionally, there is not much labelled/annotated data on this category. It is challenging to label hundreds of thousands of examples, but relatively simple to label around 100 examples. If we could design a model that is able to learn from a few examples to best predict all new incoming data, then that would solve this problem. Academically, the problem of learning complicated sentimental nuance from few examples is an interesting research problem. The hope is that the meta learning setup improves generalisation ability; the model has learned to learn from a few examples, meaning it better generalises onto the new aspect category.

Concurrently with this thesis, this idea of few shot text classification with pre-initialised language models was independently replicated by researchers at Alibaba specifically for the case of BERT [64]. They work on an already prepared meta learning text dataset, whereas we provide utilities to transform any dataset into this meta learning framework in our library, but report better results than applying MAML on its own to the randomly

[15]Indeed in the SemEval 2014 task, where we focused on Restaurant datasets, there was another dataset released that focused on Laptop reviews, where the categories were e.g battery and screen, irrelevant in the context of restaurants

initialised language model, thereby validating the usage of language model initialisation for a "better" starting point for training.

In our work, we use Reptile [60] to give a feasable methodology for training in the few shot text classification regime: For each task, which for us now is defined by a unique emotion category for the commerical dataset, or a unique aspect in the SemEval dataset, we take K positive examples of this task as well as K negative examples of this task. We do this for every category. Then, we construct mini datasets considering of $2K$ examples for each of the C categories (e.g. $C = 34$ in the case of the Streetbees Data). At training time, we perform a series of meta training steps, which consist of passing each of these mini batches of category specific datasets through the model, observing and backpropagating the loss for each category and storing all of these weight updates on each of the inner steps, then taking an outer meta training gradient descent step as per Algorithm 2 in Section 2.5. We found in our work that the gradient update rules were fragile and possibly mathematically unstable and thus the language models did not learn effectively. A more thorough investigation of learning rates and the code would have to be conducted to continue this extension, but we provide a baseline implementation that runs end-to-end in the `TraMML` library. For more information, see Appendix C.

5.4 Knowledge Graph Extensions

The company is currently working on a NER tagger, which is able to tag products and brands. This could be used in conjunction with the TABSA style setup: for each sentence, once targets t have been identified, we could prepare auxilliary questions asking about a fixed aspect set $\mathcal{A}$ about each of the targets and build a knowledge graph using the methodology outlined in this paper.

Chapter 6

Conclusion

The contributions of this thesis are multiple: we gave a thorough and critical investigation of multitask learning using language models, as well as defining and experimented with novel sampling schemas that mimic human task solving methodologies which provably increase performance on difficult downstream language tasks, in particular ABSA tasks. We explored how auxilliary tasks can act as effective regularisers, and touched on some important discussion regarding how the pretraining of these language models (in particular the inclusion of a NLI task) effects downstream performance.

We conclude that sampling proportional to a subtask hierarchy is useful in achieving SOTA results on difficult downstream tasks, but these improvements are often small in terms of their relative increase. They might be useful in certain environments where the precision and/or recall of a system is important for an application, but the main benefits almost certainly reside in the quality of a base language model. To that end, we empirically show the importance of multitask pretraining objectives in language models, showing that despite XLNet's superiority on a range of NLP tasks such as the GLUE dataset, it fails to show the same performance increases over BERT for more complicated QA/NLI downstream tasks. Thus our recommendations are twofold:

1. The community needs a tougher benchmark for Language Model evaluation in terms of testing models on (a set of) more complicated downstream tasks

2. The community should further research the interplay between pretraining tasks and downstream performance on related downstream tasks. A good starting point is a full ablation study of ERNIE 2.0 [56]

We agree with the literature in that multitask fine tuning enables the bottle to learn better, more regularised word representations allowing it to generalise better, and contribute a well commented, scalable library for continued experimentation by the research community into Multitask and Meta Learning for Language Models. We feel that this is important: the rapid rate of NLP progress leads to many new ideas being generated simultaneously, and the field would benefit from some thorough reviews and investigations into how best these transformer models learn as to direct future research. We hope this investigation provides a starting point for future work.

In terms of a corporate environment, the amount of training time required as well as time taken to build these multitask systems might offer a negligible return compared to standard transfer learning/fine tuning approaches. Thus, while our approach has proved successful, the uptake in industry for multitask systems might be slow, although useful if the upmost accuracies are required. In terms of academic contexts, this investigation provides insight into how we can manipulate the representations of language models even after pretraining to better perform at tasks, and since the idea is inspired by humanity inspired pedagogical methods this suggests that these models can benefit from learning in similar ways that humans do.

An important challenge in this space is how we generalise to new label categories, and so we outline a potential way to leverage cutting edge meta learning algorithms in order to learn more task generalisable word embeddings.

Appendix A

Streetbees: A Company Profile & Project Relevancy

Streetbees is a start-up focused on market research for large brands, enabling them to get "closer" to their consumer by conducting "in-the-moment" surveys via a conversational chatbot-like app. These surveys are either generic, or client led, and ask users questions such as "What are you eating?" and "What do you like about the product?". By answering these surveys the users, referred to as *bees*, get financial compensation.

The type of market research differs from traditional market research in two main ways: it asks for user feedback *in the moment* i.e as they are consuming or buying the product (or immediately after), and ask about contextual factors that might motivate such a consumption or purchase; and they accept *open text* responses to the questions, unlocking a potentially much richer data source than the constraints imposed by traditional multiple choice answers.

Since the company was founded in 2015, they have amassed a large dataset of close to 3 billion unique datapoints, in over 12 geographies, that capture human interaction with consumer products in the market, but also, more profoundly, form a representation of human behaviour: the underlying contextual motivating factors between purchasing decisions and buying habits. This enables them to give insights to their consumers, such as identifying previously undescovered market segmentations via clustering like the identification of a "gamer snack" group in America - a large selection of the snack market that was previously untapped, a Streetbees survey identified the market requirement for healthy, non greasy foods compatible with gamers.

The company intends to build a knowledge graph, representing the relationship between brands, their product portfolios and humans where the nodes are user "logs" and the relations include links to the product (and brand) nodes mentioned in the log. Additional attributes will include how the person was feeling (sentiment) when using/consuming said product, and motivating reasons why. With attributes like this, we can perform powerful link prediction on this graph across contextual elements, contrary to the traditional demographic factor approach.

This project directly relates to the sentiment relations/nodes of the graph, and generalises the notion of how the person was feeling overall during the log to specifically. The

methodologies developed in this thesis couple with this goal since it provides a robust framework for the improvement of the ABSA task, enabling knowledge graphs to be build that correctly identify user sentiment on a more granular level, specifically regarding particular aspects of products. The hope is that these models can soon generalise to entirely new, client driven, aspects as previously unseen during the training phase. This is a novel application of cutting edge NLP that is explored in this document.

The company has applied this NLP technology in a variety of applications: Mood prediction, Food prediction **talk more here? Move to beginning of thesis?**

Appendix B

Additional Findings

Appendix C

Code listings

Some code here or something

References

General

[1] Ronan Collobert et al. *Natural Language Processing (almost) from Scratch*. Tech. rep. arXiv: 1103.0398v1. URL: https://arxiv.org/pdf/1103.0398.pdf.

[2] Ning Kang et al. "Using rule-based natural language processing to improve disease normalization in biomedical text". In: *Journal of the American Medical Informatics Association* 20.5 (Sept. 2013), pp. 876–881. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2012-001173. URL: https://academic.oup.com/jamia/article-lookup/doi/10.1136/amiajnl-2012-001173.

[3] Eric Brill. *A SIMPLE RULE-BASED PART OF SPEECH TAGGER*. Tech. rep. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a460532.pdf.

[4] Laura Chiticariu, Yunyao Li, and Frederick R Reiss. *Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems!* Tech. rep. 2013, pp. 18–21. URL: https://www.aclweb.org/anthology/D13-1079.

[5] Gaganpreet Kaur. "USAGE OF REGULAR EXPRESSIONS IN NLP". In: *International Journal of Research in Engineering and Technology* 03.01 (Jan. 2014), pp. 168–174. DOI: 10.15623/ijret.2014.0301026. URL: https://ijret.org/volumes/2014v03/i01/IJRET20140301026.pdf.

[6] Mark-Jan Nederhof et al. *Left-To-Right Parsing and Bilexical Context-Free Grammars*. Tech. rep. URL: https://www.aclweb.org/anthology/A00-2036.

[7] Ian Tenney et al. "What do you learn from context? Probing for sentence structure in contextualized word representations". In: (May 2019). arXiv: 1905.06316. URL: http://arxiv.org/abs/1905.06316.

[8] Nils J Nilsson. *THE QUEST FOR ARTIFICIAL INTELLIGENCE A HISTORY OF IDEAS AND ACHIEVEMENTS*. Tech. rep. URL: http://www.cambridge.org/us/0521122937http://www.cambridge.org/us/0521122937http://www.cambridge.org/us/0521122937.

[9] Zellig S Harris. "Distributional Structure". In: *Distributional Structure, WORD* 10.3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520. URL: https://www.tandfonline.com/action/journalInformation?journalCode=rwrd20.

[11] Peter F Brown et al. *Class-Based n-gram Models of Natural Language*. Tech. rep. URL: https://www.aclweb.org/anthology/J92-4003.

[12] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. Tech. rep. URL: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[13] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: (Jan. 2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. *GloVe: Global Vectors for Word Representation*. Tech. rep. URL: https://nlp.stanford.edu/projects/glove/.

[15] Laurens Van Der Maaten and Geoffrey Hinton. *Visualizing Data using t-SNE*. Tech. rep. 2008, pp. 2579–2605. URL: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[16] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (Mar. 1951), pp. 79–86. ISSN: 0003-4851. DOI: 10.1214/aoms/1177729694. URL: http://projecteuclid.org/euclid.aoms/1177729694.

[17] Jeffreyl Elman. *Finding Structure in Time*. Tech. rep. 1990, pp. 179–211. URL: https://pdfs.semanticscholar.org/b71e/c700edfec2b969c9d27d33eac09188290294.pdf.

[18] Paul J Werbos. *Backpropagation Through Time: What It Does and How to Do It*. Tech. rep. URL: http://www.cs.cmu.edu/%7B~%7Dbhiksha/courses/deeplearning/Fall.2016/pdfs/Werbos.backprop.pdf.

[19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. Tech. rep. arXiv: 1211.5063v2. URL: https://arxiv.org/pdf/1211.5063.pdf.

[20] Sepp Hochreiter and J J Urgen Schmidhuber. *Long Short-Term Memory*. Tech. rep. 8. 1997, pp. 1735–1780. URL: http://www7.informatik.tu-muenchen.de/%7B~%7Dhochreithttp://www.idsia.ch/%7B~%7Djuergen.

[21] Savvas Varsamopoulos, Koen Bertels, and Carmen G. Almudever. "Designing neural network based decoders for surface codes". In: (Nov. 2018). arXiv: 1811.12456. URL: http://arxiv.org/abs/1811.12456.

[22] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. *SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS*. 2013. ISBN: 9781479903566. URL: https://wiki.inf.ed.ac.uk/twiki/pub/CSTR/ListenSemester2201213/graves-icassp2013.pdf.

[23] Matthew E. Peters et al. "Deep contextualized word representations". In: (Feb. 2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[24] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. Tech. rep. URL: https://gluebenchmark.com/leaderboard.

[25] Ashish Vaswani et al. *Attention Is All You Need*. Tech. rep. arXiv: 1706.03762v5. URL: https://arxiv.org/pdf/1706.03762.pdf.

[26] Zihang Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: (Jan. 2019). arXiv: 1901.02860. URL: http://arxiv.org/abs/1901.02860.

[27] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[28] Gao Huang et al. "Densely Connected Convolutional Networks". In: (Aug. 2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[29] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009, pp. 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848. URL: https://ieeexplore.ieee.org/document/5206848/.

[30] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. Tech. rep. arXiv: 1405.0312v3. URL: https://arxiv.org/pdf/1405.0312.pdf.

[31] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: (Jan. 2018). arXiv: 1801.06146. URL: http://arxiv.org/abs/1801.06146.

[33] Jianpeng Cheng, Li Dong, and Mirella Lapata. "Long Short-Term Memory-Networks for Machine Reading". In: (Jan. 2016). arXiv: 1601.06733. URL: http://arxiv.org/abs/1601.06733.

[34] Ankur Parikh et al. *A Decomposable Attention Model for Natural Language Inference*. Tech. rep. URL: https://aclweb.org/anthology/D16-1244.

[35] Romain Paulus, Caiming Xiong, and Richard Socher. "A Deep Reinforced Model for Abstractive Summarization". In: (May 2017). arXiv: 1705.04304. URL: http://arxiv.org/abs/1705.04304.

[36] Zhouhan Lin et al. "A Structured Self-attentive Sentence Embedding". In: (Mar. 2017). arXiv: 1703.03130. URL: http://arxiv.org/abs/1703.03130.

[37] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: (July 2016). arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450.

[38] Sarthak Jain and Byron C Wallace. *Attention is not Explanation*. Tech. rep. arXiv: 1902.10186v3. URL: https://github.com/successar/.

[39] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (Oct. 2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[40] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: (Apr. 2018). arXiv: 1804.07461. URL: http://arxiv.org/abs/1804.07461.

[41] Wilson L. Taylor. ""Cloze Procedure": A New Tool for Measuring Readability". In: *Journalism Bulletin* 30.4 (Sept. 1953), pp. 415–433. ISSN: 0197-2448. DOI: 10.1177/107769905303000401. URL: http://journals.sagepub.com/doi/10.1177/107769905303000401.

[42] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (July 2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[43] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. Tech. rep. 2016. arXiv: 1609.08144v2. URL: https://arxiv.org/pdf/1609.08144.pdf.

[44] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: (June 2019). arXiv: 1906.08237. URL: http://arxiv.org/abs/1906.08237.

[45] XLNet Team. *A Fair Comparison Study of XLNet and BERT with Large Models*. 2019. URL: https://medium.com/@xlnet.team/a-fair-comparison-study-of-xlnet-and-bert-with-large-models-5a4257f59dc0 (visited on 08/19/2019).

[46] Ian Tenney, Dipanjan Das, and Ellie Pavlick. "BERT Rediscovers the Classical NLP Pipeline". In: (May 2019). arXiv: 1905.05950. URL: http://arxiv.org/abs/1905.05950.

[47] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. "Learning to Generate Reviews and Discovering Sentiment". In: (Apr. 2017). arXiv: 1704.01444. URL: http://arxiv.org/abs/1704.01444.

[48] Richard Socher et al. *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. Tech. rep. URL: http://nlp.stanford.edu/.

[49] Svetlana Kiritchenko et al. *NRC-Canada-2014: Detecting Aspects and Sentiment in Customer Reviews*. Tech. rep. 2014, pp. 437–442. URL: http://nlp.stanford.edu/software/corenlp.shtml.

[50] Chi Sun, Luyao Huang, and Xipeng Qiu. *Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence*. Tech. rep. 2019. arXiv: 1903.09588v1.

[57] Ryan Garland, Sophia Goldberg, and Erik Mathiesen (Streetbees). "Embedding Knowledge Graphs as Languages with BERT". In: *EurNLP*. 2019.

[58] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. *A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications*. Tech. rep. 2017, p. 1. arXiv: 1709.07604v3. URL: https://arxiv.org/pdf/1709.07604.pdf.

[65] Erik F Tjong, Kim Sang, and Fien De Meulder. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. Tech. rep. URL: http://lcg-www.uia.ac.be/conll2003/ner/.

[66] Google and Lanpa. *Tensorboard & TensorboardX*. 2019. URL: https://github.com/lanpa/tensorboardX.

[67] Volodymyr Kuleshov and Doina Precup. *Algorithms for the multi-armed bandit problem.* Tech. rep. 2000, pp. 1–48. arXiv: 1402.6028v1. URL: https://arxiv.org/pdf/1402.6028.pdf.

[68] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision.* Tech. rep. arXiv: 1512.00567v3. URL: https://arxiv.org/pdf/1512.00567.pdf.

Multitask Learning

[10] Rich Caruana. *Multitask Learning *.* Tech. rep. 1997, pp. 41–75. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.8707%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[32] Sebastian Ruder. "Neural Transfer Learning for Natural Language Processing". PhD thesis. National University of Ireland, 2019.

[51] German I. Parisi et al. "Continual Lifelong Learning with Neural Networks: A Review". In: (Feb. 2018). DOI: 10.1016/j.neunet.2019.01.012.. arXiv: 1802.07569. URL: http://arxiv.org/abs/1802.07569%20http://dx.doi.org/10.1016/j.neunet.2019.01.012..

[52] Sebastian Ruder. "An Overview of Multi-Task Learning in Deep Neural Networks". In: (June 2017). arXiv: 1706.05098. URL: http://arxiv.org/abs/1706.05098.

[53] Jonathan Baxter. "A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling". In: *Machine Learning* 28.1 (1997), pp. 7–39. ISSN: 08856125. DOI: 10.1023/A:1007327622663. URL: http://link.springer.com/10.1023/A:1007327622663.

[54] Asa Cooper Stickland and Iain Murray. "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning". In: (Feb. 2019). arXiv: 1902.02671. URL: http://arxiv.org/abs/1902.02671.

[55] Victor Sanh, Thomas Wolf, and Sebastian Ruder. "A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks". In: (Nov. 2018). arXiv: 1811.06031. URL: http://arxiv.org/abs/1811.06031.

[56] Yu Sun et al. "ERNIE 2.0: A Continual Pre-training Framework for Language Understanding". In: (July 2019). arXiv: 1907.12412. URL: http://arxiv.org/abs/1907.12412.

Meta Learning

[59] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: (Mar. 2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[60] Alex Nichol, Joshua Achiam, and John Schulman Openai. *On First-Order Meta-Learning Algorithms.* Tech. rep. arXiv: 1803.02999v3. URL: https://arxiv.org/pdf/1803.02999.pdf.

[61] Thomas Wolf, Julien Chaumond, and Clement Delangue. *Workshop track-ICLR 2018 META-LEARNING A DYNAMICAL LANGUAGE MODEL*. Tech. rep. URL: https://openreview.net/pdf?id=SyikZmkDM.

[62] Junkun Chen et al. "Meta multi-task learning for sequence modeling". In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 5070–5077.

[63] Jiatao Gu et al. *Meta-Learning for Low-Resource Neural Machine Translation*. Tech. rep., pp. 3622–3631. URL: https://aclweb.org/anthology/D18-1398.

[64] Mo Yu et al. *Diverse Few-Shot Text Classification with Multiple Metrics*. Tech. rep., pp. 1206–1215. URL: https://github.com/Gorov/DiverseFewShot%7B%5C_%7DAmazon.