Improving Generalization in Natural Language Inference by Joint Training with Semantic Role Labeling

by

Cemil Cengiz

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

June 9, 2020

Improving Generalization in Natural Language Inference by Joint Training with Semantic Role Labeling

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Cemil Cengiz

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Deniz Yuret

_____

Assoc. Prof. Arzucan Özgür

_____

Prof. Engin Erzin

Date: _____

To my beloved mom

ABSTRACT

Improving Generalization in Natural Language Inference by Joint Training with Semantic Role Labeling

Cemil Cengiz

Master of Science in Computer Science and Engineering

June 9, 2020

Recently, end-to-end models have achieved near-human performance on natural language inference (NLI) datasets. However, they show low generalization on out-of-distribution evaluation sets since they tend to learn shallow heuristics due to the biases in the training datasets. The performance decreases dramatically on diagnostic sets measuring compositionality or robustness against simple heuristics. Existing solutions for this problem employ dataset augmentation by extending the training dataset with examples from the evaluated adversarial categories. However, that approach has the drawbacks of being applicable to only a limited set of adversaries and at worst hurting the model performance on other adversaries not included in the augmentation set. Instead, our proposed solution is to improve sentence understanding (hence out-of-distribution generalization) with joint learning of explicit semantics. In this thesis, we show that a BERT based model trained jointly on English semantic role labeling (SRL) and NLI achieves significantly higher performance on external evaluation sets measuring generalization performance.

ÖZETÇE

Yüksek Lisans Tez Başlığı
Cemil Cengiz
Bilgisayar Mühendisliği, Yüksek Lisans
9 Haziran 2020

Son zamanlarda, uçtan uca modeller, doğal dil çıkarımı (NLI) veri kümelerinde insan seviyesine yakın bir performans sergilemiştir. Bununla birlikte, eğitim veri setlerindeki meyiller nedeniyle sığ sezgisellikler öğrenme eğiliminde oldukları için dağıtım dışı değerlendirme setlerinde düşük genelleme gösteriyorlar. Bir araya getirilebilirlik veya basit sezgiselliklere karşı dayanıklılığı ölçen tanı kümelerinde performans önemli ölçüde düşmektedir. Bu soruna yönelik mevcut çözümlerde, eğitim veri kümesini, değerlendirilen çekişmeli kategorilerden örneklerle genişleterek veri kümesi genişletmesi kullanılmaktadır. Fakat, bu yaklaşımın sadece sınırlı bir dizi çekişmeli sınıf için geçerli olmasının yanı sıra, en kötü ihtimalde genişletme setinde yer almayan diğer çekişmeli örnekler üzerindeki performansı zedelemesi gibi dezavantajları bulunmaktadır. Bunun yerine önerdiğimiz çözüm, açık olarak anlamsallığın ortak öğrenimi ile cümle anlayışını (dolayısıyla dağıtım dışı genellemeyi) geliştirmektir. Bu tezde, İngilizce anlamsal rol etiketleme (SRL) ve NLI ile ortaklaşa eğitilen BERT tabanlı bir modelin, genelleme performansını ölçen dış değerlendirme setlerinde önemli ölçüde daha yüksek performans elde ettiğini gösteriyoruz.

ACKNOWLEDGMENTS

I consider myself very lucky for having the chance of studying with my advisor, Deniz Yuret. I learned from him the importance of always trying to follow a systematic approach, not to accept the results without questioning them, and not to give up easily when things go awry. I am also grateful to Engin Erzin and Arzucan Özgür for participating in my thesis committee.

I would like to thank my colleagues at AI Lab, especially Ulaş Sert and Berkay Önder, both of whom I learned a lot from, and of whose friendship made my lab experience much more enjoyable.

I want to thank Emre Köse, Ömer Yaman, Beakal Gizachew Assefaa and Serhat Can Kadıoğlu for the great time and valuable discussions we shared.

I acknowledge the financial support for my master's studies provided by Koç University, Technological Research Council of Turkey (TÜBİTAK) and Huawei Turkey R&D Center.

Finally, I am very grateful to my family. I could not have achieved this without their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NLI	Natural Language Inference
SRL	Semantic Role Labeling
RNN	Recurrent Neural Networks
BERT	Bidirectional Encoder Representations from Transformers

Chapter 1

INTRODUCTION

Natural language inference (NLI) is about determining the inference relationship between two sentences. This task is also known as recognizing textual entailment (RTE), but NLI term has been much more frequently used recently. Concretely, given a premise sentence p, and a hypothesis sentence h, NLI is the task of determining the inference relationship from p to h. It requires understanding the individual meaning of the premise and hypothesis sentences as well as deciding the correct inference relationship between them. It is usually formulated as a three-class classification task with *entailment*, *contradiction* and *neutral* labels. However, there are some NLI variants with binary labels consisting of single negative and positive labels such as the RTE dataset in GLUE benchmark [Wang et al., 2018]. The general motivation behind using binary labels is that sometimes it is hard to discriminate the *contradiction* label from *neutral*, so this type of ambiguous examples can be avoided by employing binary labels.

In previous research, sequence encoders connected to a classifier head have been commonly used as NLI systems [Conneau et al., 2017b]. Traditionally, the encoder layer has been an RNN-based model such as LSTM [Hochreiter and Schmidhuber, 1997]. [Vaswani et al., 2017] proposed the Transformer as an alternative model to the RNN. Since the Transformer is based on a self-attention mechanism rather than recurrent layers, it is much faster to train in parallel and can capture distant dependencies better. Therefore, the recent models originated from Transformer replaced the RNN-based encoders in many systems trained for natural language understanding tasks such as NLI, Question Answering, Common Sense Reasoning [Radford et al., 2018], and Neural Machine Translation [Lakew et al., 2018]. As

a Transformer based model, BERT uses self-attention to capture the relationships within the input text during encoding, which can include one or more sentences [Devlin et al., 2018]. Therefore, it can learn a joint representation for a premise-hypothesis pair, which can be fed to a classifier layer to predict the inference relation between them.

To succeed in NLI, a system must have strong reasoning skills and a good understanding of the language [MacCartney, 2009]. If a large annotated dataset is available, the system can be trained from scratch for NLI, learning both the language and reasoning simultaneously. However, such data is often not available. Without seeing many syntactic variation and inference relation combinations, learning both is a hard task. Therefore, separating the two by training a language model first, and adjusting it for NLI later can be a more effective approach. Due to the development of powerful language models that can be trained on unlabeled data in an unsupervised manner, many pre-trained context-aware encoders such as BERT [Devlin et al., 2018] and GPT [Radford et al., 2018] are publicly available. Noticeably, [Devlin et al., 2018] showed that BERT can be effectively used on many natural language understanding tasks, including NLI. Combining a pre-trained BERT encoder with a task-specific head, and then fine-tuning the entire model on the target task achieved state-of-the-art results on a number of tasks.

NLI has been regarded as a central problem in natural language understanding and found its place in benchmarks such as GLUE [Wang et al., 2018]. Contemporary neural network based models achieve state-of-the-art (SOTA) results on these benchmarks. However, scoring high on a test set that has a similar distribution to the training set does not guarantee wider generalization. Models that top the leaderboards on standard test sets may perform poorly on specifically constructed evaluation sets targeting dataset biases. For instance, the HANS challenge dataset [McCoy et al., 2019b] showed that models trained on NLI get fooled easily by heuristics when the input sentence pairs have high lexical similarity. The following example from HANS can explain how this might happen.

Premise: *The judge by the actor stopped the banker.*

Hypothesis: *The banker stopped the actor.*

Premise	The judge by the actor stopped the banker.
VERB	stopped
ARG0	The judge by the actor
ARG1	the banker
Hypoth.	The banker stopped the actor.
VERB	stopped
ARG0	The banker
ARG1	the actor

Table 1.1: An example pair from HANS, including the semantic roles of the words in each sentence. *ARG0* represents the proto-agent, i.e. the thing that stops, *ARG1* represents the proto-patient, i.e. the object being stopped, in this example.

A human reading this sentence pair carefully can conclude that the hypothesis can not be inferred from the premise. However, models relying on the lexical overlap heuristic will be fooled and predict the label as *entailment* since the premise contains all words of the hypothesis. Existing approaches commonly tackle such adversaries by training the model with a dataset augmented with similar adversarial examples. As detailed in Section 5.3, the problem with this approach is that it might lead to overfitting to the adversaries on the augmentation set. Therefore, it can decrease the performance on other possible adversaries and hurt the generalization ability of the model [Nie et al., 2018].

Semantic Role Labeling (SRL) asks the *"who did what to whom, when and where etc."* questions to find the semantic roles of words or phrases in a sentence [He et al., 2017]. We hypothesize that using the SRL task as a joint objective should improve the semantic knowledge of the models, thus making them less prone to dataset biases. Consider the semantic roles in the previous example sentence pair, which are shown in Table 1.1. We can see that the role of *"the banker"* differs between the sentences. Since *"stop"* is not a reciprocal verb, a model that is aware of the

semantic roles can find out that the inference relation is *non-entailment* although the premise contains all words in the hypothesis, albeit with a different order. In contrast, if a model pays too much attention to lexical similarity, it might falsely predict the relation as *entailment* as the SOTA models analyzed on HANS such as BERT [Devlin et al., 2018] do. SRL informs the model directly about the semantic roles of words which makes it easier for the model to rely on more than just shallow lexical cues.

Our contribution in this thesis is threefold:

- We propose a BERT based multi-task learning (MTL) model jointly trained on English SRL and NLI (Section 3.3), and show that this model achieves scores comparable to the single-task BERT on both tasks (Chapter 4).

- We evaluate the proposed model on out-of-distribution test sets such as HANS [McCoy et al., 2019b] and Comparisons [Dasgupta et al., 2018] and demonstrate that it exceeds the single-task BERT performance significantly. Specifically, when trained on MultiNLI, the multi-task model exceeds the single-task model accuracy by 4% on HANS and 5.3% on Comparisons without using data augmentation. (Section 4.1.2 and 4.2)

- We compare the proposed MTL approach to the sequential transfer learning and show that the MTL is more helpful. (Section 4.1.3)

The structure of this thesis is organized as follows:

Chapter 2 explains the datasets used in the experiments.

Chapter 3 describes our model starting from a preliminary architecture, continuing with the baseline and finally presenting its multi-task learning variant.

Chapter 4 describes the experiments we conducted and their results.

Chapter 5 outlines the related work in the literature.

Chapter 6 concludes this thesis and presents future research directions.

Chapter 2

DATASETS

This chapter describes the datasets used in the experiments. We explain the shortcomings of the existing NLI datasets and describe an SRL dataset that can be used to alleviate these shortcomings.

2.1 Large-scale NLI Datasets

In this work, we consider two open-domain, large-scale NLI datasets in English, SNLI [Bowman et al., 2015] and MultiNLI [Williams et al., 2018]. SNLI is created using image captions written by humans whereas MultiNLI includes five different genres of written and spoken English such as telephone conversations, travel guides and press releases from government websites. A detailed comparison of these datasets can be found in Table 2.1. Both datasets have been used for training general NLI models, or as an intermediate training resource for transfer learning to a domain-specific dataset, possibly with smaller size [Cengiz et al., 2019].

Dataset	**Genre**	**Training Set Size**
MutliNLI	Fiction	77,348
	Government	77,350
	Slate	77,306
	Telephone	83,348
	Travel	77,350
	Total	392,702
SNLI	Image Captions	550,152

Table 2.1: Comparison of NLI datasets by their genres and sizes.

Recently, deep neural network models achieved human-level performance on NLI tasks in benchmarks such as GLUE [Wang et al., 2018] and SuperGLUE [Wang et al., 2019]. However, as [McCoy et al., 2019b] and [Dasgupta et al., 2018] showed, these results do not reflect the performance of the models on out-of-distribution test sets. They proposed such adversarial test sets targeting specific biases apparent in the original NLI datasets to show that SOTA models are vulnerable to these superficial patterns.

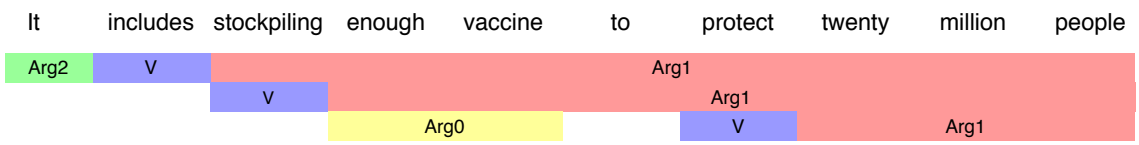
2.2 Adversarial NLI Datasets

We present two adversarial NLI datasets proposed to evaluate the models trained on NLI datasets for reliability against some heuristics present in the training data. The first one, HANS, targets the models trained on MultiNLI whereas the second one, Comparisons, was originally proposed to evaluate the models trained on SNLI.

HANS is an extensive evaluation set proposed by [McCoy et al., 2019b], that consists of three types of adversarial examples: *lexical overlap*, *subsequence* and *constituent*. *Lexical overlap* is the most general category, indicating all the words in the hypothesis sentence are present in the premise as well. An example from this category with *entailment* gold label is the following: *"The banker near the judge saw the actor. ⇒ The banker saw the actor."* *Subsequence* is a special case of *lexical overlap*, indicating that the hypothesis is a contiguous subsequence of the premise. The following pair is an example from this category with *entailment* gold label: *"The artist and the student called the judge. ⇒ The student called the judge."* Lastly, *constituent* is a special case of the *subsequence*, denoting that the hypothesis is a complete subtree of the premise's constituency parse tree. An instance from the *constituent* category with *non-entailment* gold label is as following: *"If the actor slept, the judge saw the artist. ⇒ The actor slept."* It is worth noting that although there is a hierarchical relation between the categories, their instances do not overlap. Therefore, they will be treated as distinct categories throughout the paper. The sentences included in this dataset were created using templates and ensured to be plausible. Moreover, the verbs were chosen from the frequently used

verbs in MultiNLI so that the models trained on MultiNLI are familiar with them. Differently than MultiNLI, this dataset has binary labels, a label is either *entailment* or *non-entailment*. Finally, this dataset has two partitions with identical size, a test set to evaluate the models and an augmentation set to augment the MultiNLI training set. In our experiments, we treat the augmentation set as a validation set and do not use it for training.

The Comparisons dataset [Dasgupta et al., 2018] attempts to evaluate the models trained on SNLI for three types of adversarial examples: *same*, *more-less*, *not*. The *same* category consists of hypothesis-premise pairs having exactly the same words in a different order. An example with *contradiction* gold label is as following: *"The woman is more cheerful than the man. ⇒ The man is more cheerful than the woman."* The *more-less* type contains instances whose sentence pair differ by including the word *"more"* or the word *"less"*, and possibly in word ordering. The following pair is an example from this category with *entailment* label: *"The woman is more cheerful than the man. ⇒ The man is less cheerful than the woman."* The third category is the *not* type, representing the instances having the negation word *"not"* either in the hypothesis or in the premise, but not in both. The word order of the sentences might differ as well. A sentence pair from this category with *entailment* gold label is the following: *"The woman is more cheerful than the man. ⇒ The man is not more cheerful than the woman."* The authors created this dataset by automatically generating examples fitting into one of the described categories using a vocabulary similar to SNLI's. Moreover, they analyzed SNLI and showed that it has many examples fitting into the examined categories with labels mostly supporting the heuristic choice. Unlike SNLI, this dataset does not have the *neutral* label. It has the *entailment* label for the positive examples and the *contradiction* label for the negative examples.



include
Arg0: agent, entity causing some grouping
Arg1: theme, thing being included in some group
Arg2: group

stockpile
Arg0: agent, storer
Arg1: commodity
Arg2: location or destination

protect
Arg0: protector
Arg1: protected
Arg2: danger
Arg3: instrument

Figure 2.1: The semantic role labels for the sentence *"It includes stockpiling enough vaccine to protect twenty million people"*. The top part shows the PropBank [Kingsbury and Palmer, 2002] predicate-argument annotations whereas the bottom part shows the frame files of the predicates.

2.3 Semantic Role Labeling as an Auxiliary Objective for Sentence Understanding

We use the English Ontonotes v5.0 SRL dataset with the CONLL-2012 shared task format [Pradhan et al., 2013] which gives the predicate-argument structure for each sentence. This dataset follows the PropBank [Kingsbury and Palmer, 2002] style for annotation. In this formulation, each predicate has a frame file denoting the argument types it can take and their meanings with explanations and examples. The arguments are labeled with ordered non-negative integers e.g. *Arg0*, *Arg1* etc. but their exact meaning may change depending on the predicate. An example sentence and its gold standard labels together with the frame files of the predicates are shown in Figure 2.1. Each word has multiple different roles attached to it with respect to a particular predicate. In this sentence, there are three verbal predicates, hence three types of labels associated with each word.

The auxiliary task we used is formulated as prediction of the arguments for a given predicate in a sentence. Therefore, each predicate in a sentence together with the semantic role label spans associated with it yield a different training instance. While training or evaluating a model, the words which do not have a relation with

the current predicate are assigned to *O* (i.e. *Other*) tag as their argument type. The number of training instances in the whole dataset is around 280,000.

Chapter 3

MODEL

We used BERT as our main model component for the NLI task. The baseline model is the single-task BERT classifier directly trained on the NLI dataset whereas our improved model is the multi-task BERT trained jointly on SRL and NLI.

The basic building blocks of BERT, i.e the encoder layers, were inspired from an effective encoder-decoder architecture which is known as the Transformer [Vaswani et al., 2017]. Specifically, BERT is constructed by stacking a number of Transformer encoder blocks sequentially. Therefore, it is crucial to understand the Transformer encoder, especially its main powerhouse which is the self-attention, in order to understand BERT. Thus, this chapter starts by reviewing the Transformer encoder. Then, it describes the BERT classifier as the baseline and finally presents the model we propose, the multi-task BERT.

3.1 Transformer Encoder

Transformer is an encoder-decoder model that was originally proposed as a machine translation system although it can be applied to various other sequence-to-sequence modeling problems. It has two main building blocks, an encoder and a decoder layer. The full model has consecutive encoder layers followed by consecutive decoder layers. BERT, in its basic form without any task-specific layer, is an encoder network constructed by stacking multiple layers of Transformer encoder sequentially. This section explains the details of the Transformer encoder architecture and omits the decoder since BERT does not use it.

A Transformer encoder is a sequence-to-sequence layer that accepts a sequence of vectors and outputs another sequence of vectors. Each vector in the sequence corresponds to the embedding of the input token at that layer. The first encoder layer

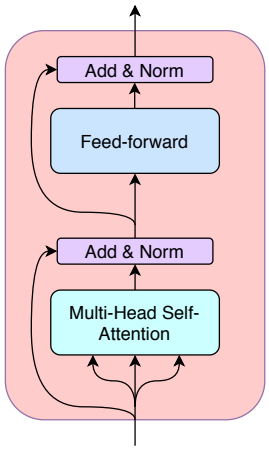


Figure 3.1: Transformer encoder layer from [Vaswani et al., 2017].

accepts the word embeddings coming from the input and emits the processed version of them to the second layer. From the second layer onwards, each layer accepts a list of token representations from the previous layer, processes them and sends the resulting sequence to the next layer. The sequence length and the dimension of each embedding vector are kept unchanged so that an arbitrary number of encoder blocks can be stacked on top of each other conveniently.

Internally, a Transformer encoder layer has two main blocks, a multi-head attention sub-layer and a feed-forward neural network sub-layer. Moreover, there is a layer-normalization [Ba et al., 2016] at the end of each sub-layers. Figure 3.1 depicts the overall encoder architecture. The overall formula for the output of each sub-layer is $LayerNorm(x + Sublayer(x))$ where $Sublayer(x)$ is the function implemented by the sub-layer itself.

3.1.1 Scaled Dot-Product Attention

The scaled dot-product attention is the self-attention mechanism utilized by the Transformer. It is used to compute the context-aware embedding of each token without relying on any recurrence or convolution operations at all. In this formulation, each token is associated with three different vectors i.e. *query*, *key* and *value*. The output vector for each token is calculated by taking the weighted average of the

value vectors of all tokens in the sequence. Those weights are calculated by computing the dot product of the *query* vector of the current token and *key* vector of all tokens in the sequence. Then, they are divided by the square root of the embedding size for scaling, and normalized with softmax function to make their sum equal to 1. Given the dimension of a *key* d_k, *key* matrix K, *query* matrix Q and *value* matrix V, the matrix of output embeddings can be calculated as:

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

Each self-attention block is called a head in the multi-head attention sub-layer of the Transformer encoder. The self-attention enables each token to be aware of all tokens in the sequence while computing its representation. Moreover, this mechanism is what makes the Transformer more efficient compared to the previous architectures. The computations can easily be implemented as matrix operations that can run fast in parallel accelerator devices such as GPU and TPU.

3.1.2 Multi-head Attention

The multi-head attention has multiple independent self-attention blocks run in parallel. Instead of applying self-attention using d_{model} dimensional *query*, *key* and *value* vectors, [Vaswani et al., 2017] came up with the idea of projecting these vectors into smaller dimensions using multiple projection matrices and compute the attention on these smaller spaces. The multi-head attention sub-layer combines the outputs of multiple self-attention heads by first concatenating them and then converting into the same dimension as the inputs using a linear transformation. This process can be summarized with the following equation:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{3.2}$$

where the output of each head is calculated as the following:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{3.3}$$

where $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ and $W^O \in R^{hd_v \times d_{model}}$ are the parameter matrices used to project the original *query*, *key* and *value* vectors

represented in matrix form. The number of attention heads (h) and the dimension of the embeddings are hyper-parameters, which are different between the Transformer and BERT. In the BERT model we use in this work, there are 12 attention heads and the embedding size d_{model} is 768. Moreover, $d_k = d_v = d_{model}/h = 64$.

Finally, the outputs are summed up with the original inputs using a residual connection and layer normalization [Ba et al., 2016] is applied to the resulting embeddings. These are the output of the multi-head attention sub-layer. Using multiple independent heads enables the model to form different relationships among the tokens using representations from different subspaces.

3.1.3 Position-wise Feed-forward Network

The token embeddings coming from the multi-head attention are fed through a feed-forward neural network separately. This network consists of two linear transformations with a ReLU activation function in between. The first linear transformation expands the dimension of the token embeddings whereas the second one shrinks them back to their original size. Finally, there is again a summation followed by layer normalization that is applied to the output of this sub-layer and its input which is carried with a residual connection.

3.1.4 Positional Encoding

Self-attention is an efficient way to compute contextual embeddings but it has the drawback of omitting the order of the input tokens. Unlike the recurrent neural networks, the sequence order is not implicitly encoded, and there is no way to know the relative or absolute position of the tokens. Therefore, to make the network aware of the ordering, we add a special type of embeddings to the word embeddings before the first encoder layer. These special vectors i.e. the positional encodings [Vaswani et al., 2017] are computed using discrete sinusoidal signals. Positional encodings are used in BERT as well although its authors chose a slightly different name for them, *position embeddings*. They have the same size as the word embeddings, which makes summing them feasible. The value of the position embeddings can be calculated with

the following equations:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{3.4}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{3.5}$$

where pos is the position and i represents the dimension.

In fact, each dimension of the position embeddings is a discrete sinusoidal signal with a different frequency. [Vaswani et al., 2017] showed that this formulation allows the model to infer the relative positions of the tokens.

3.2 Single-task BERT Classifier for NLI

Our baseline model is a BERT encoder [Devlin et al., 2018] combined with a classification head. The head outputs probabilities from a three-way softmax corresponding to the three possible labels a sentence pair can have. The overview of this archi-

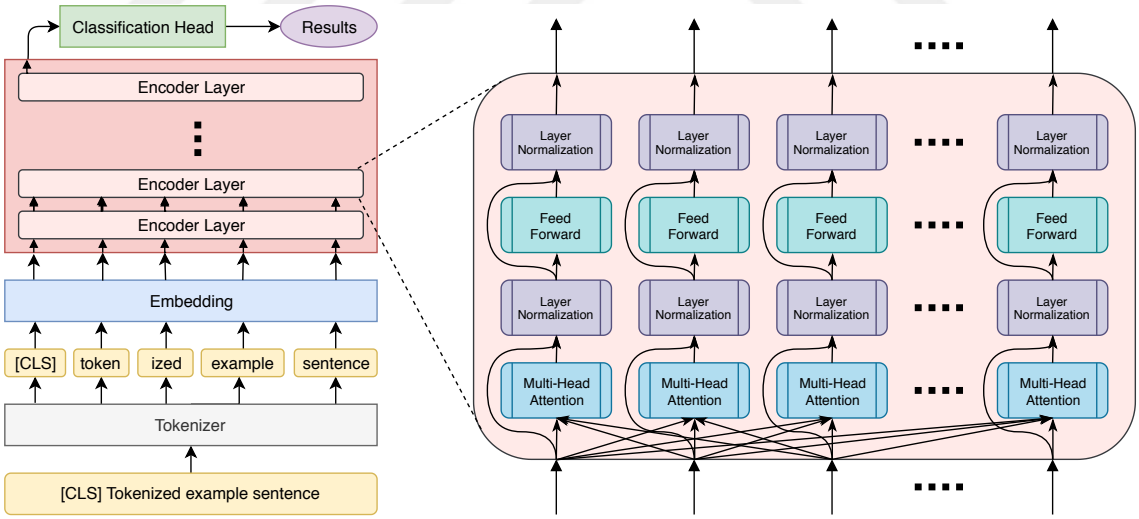


Figure 3.2: Baseline model architecture. On the left is the overview of the model, which includes the tokenizer, embedding layer, BERT encoder, and the classification head. The BERT encoder consists of twelve encoder layers. On the right are the details of what an encoder layer consists of. Each input and output of an encoder layer corresponds to a single token. Modules that are side-by-side share the same weights, only differing in inputs.

tecture can be seen from Figure 3.2. Note that this figure shows the classification procedure when the input is a single sentence. However, it is easy to use this model for sentence pair classification by appending the sentences next to each other. For that to work, BERT needs to know where each sentence ends and the next sentence begins, so we append a special [SEP] token to the end of each sentence to indicate their end positions as will be shown later.

3.2.1 BERT Encoder

We use BERT to encode the input sentence pairs. Utilizing Transformer layers and self-attention [Vaswani et al., 2017], BERT looks at how the tokens are related and outputs a hidden vector for each token inside the input sequence. Therefore, BERT can process the sentence pair together as a whole sequence and output the encoded representation for all of it in one pass.

One of the reasons why we opted for a model like the BERT encoder is that it completely avoids relying on recurrence and convolution operations. Replacing those with simple operations of self-attention (e.g. plain matrix multiplications) makes it more parallelizable and thus faster to train. Another reason is the strong starting point of BERT. It is a language modeling architecture, successfully trained on massive corpora. There are pre-trained weights available which can be fine-tuned on the down-stream task in a relatively short time.

3.2.2 Pre-Trained Weights

The original paper from [Devlin et al., 2018] presented two different pre-trained weights, BERT_{BASE} and BERT_{LARGE}. Both models were trained from scratch on English Wikipedia articles and books, and have the same vocabulary. However, the number of attention heads and the size of the feed-forward network in an encoder layer, and the total number of encoder layers are much higher in BERT_{LARGE} model. BERT_{LARGE} has 340 million parameters whereas BERT_{BASE} has 110 million parameters. The large model got slightly larger scores across a range of tasks analyzed by [Devlin et al., 2018]. However, the difference in the size of these weights result

"It includes stockpiling enough vaccine."
["it", "includes", "stock", "##pi", "##ling", "enough", "vaccine", "."]

Figure 3.3: WordPiece tokenization example. The original sentence is shown on top whereas its tokenized version is shown below as a list of wordpiece tokens.

in drastic change in their training time and memory requirements. Since we could pick larger mini-batch sizes and the training is much faster, we used the BERT_{BASE} version. Another advantage of this choice is that we could directly compare our results with the official BERT scores on HANS published by [McCoy et al., 2019b] since they used BERT_{BASE} as well.

3.2.3 Tokenization and Input Representation

BERT uses WordPiece tokenizer [Wu et al., 2016], which divides the words into smaller sub-word units called wordpieces. The vocabulary i.e. the collection of the wordpieces are constructed from the frequent wordpieces in the training corpus. In the case of BERT, the wordpiece tokenizer is trained on the corpora on which the BERT itself is trained as a language model. While processing a given word, the tokenizer breaks it into wordpieces using the trained wordpiece model. An example tokenization using the trained BERT WordPiece tokenizer is shown in Table 3.3. Apparently, *"stockpiling"* is a rare word since it is split into sub-words whereas the remaining words are common words in the training corpus as they are left unchanged. Additionally, the letters are uncapitalized as we used the uncased version of the tokenizer.

The advantage of this tokenizer is that it maintains a good compromise between the character based representation's flexibility and the word based representation's efficiency. This balance improves the overall accuracy of the natural language system [Wu et al., 2016]. Moreover, since it splits the infrequent words into wordpieces, it naturally handles the rare words problem.

During its pre-training process, one of the tasks BERT has to learn is the next sentence prediction [Devlin et al., 2018]. It is a two-sentence classification task which requires predicting if the given sentences follow each other in the original text. Because our task is also a two-sentence classification task, we mimicked the inputs BERT receives during the pre-training. Thus, our input sentence pair is represented as "[*CLS*] *Premise* [*SEP*] *Hypothesis* [*SEP*]". The [*CLS*] and [*SEP*] are special tokens, denoting "Classification" and "Separator" respectively. Since those are the tokens used during pre-training, they are kept in the same format to fully utilize the encoder's understanding of sentence pairs. Figure 3.4 shows the token embedding for an example sentence pair. As shown by the figure, another important detail while using the BERT encoder for a sentence pair task is the segment embeddings. These embeddings are used to denote which sentence a token belongs to, and learned with the other model parameters during training. While generating the input embeddings before the first encoder layer, we sum the token embeddings with the segment embeddings in addition to the position embeddings.

3.2.4 Classification Head

To transform the token representations obtained from the encoder into label predictions, we use a small classification head. Following the convention of [Devlin et al., 2018], we use the representation of the first token, [*CLS*], as the summary of

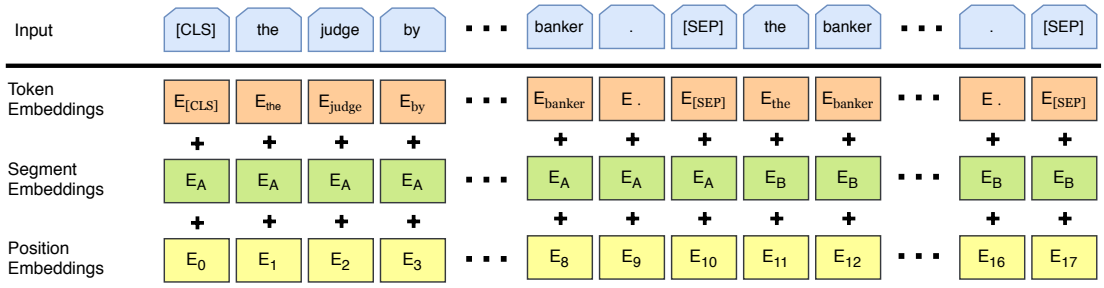


Figure 3.4: Input representation of *"The judge by the actor stopped the banker. ⇒ The banker stopped the actor."* sentence pair for NLI task. Both the single-task and multi-task models use this formulation.

the whole sequence. Then, this vector is linearly projected into a three-dimensional space such that each dimension represents the score for a label. Finally, a softmax operation is applied to convert the scores into class probabilities.

3.3 Multi-task BERT for Joint Training on SRL and NLI

In this section, we present our novel multi-task BERT model to jointly predict semantic roles and perform natural language inference [Cengiz and Yuret, 2020]. BERT is used as the shared encoder module and two separate decoder heads are appended on top of it to perform task-specific operations. The overall picture of the model can be seen from Figure 3.5. Following [Liu et al., 2019], the tasks share the encoder part of the model including the initial word embedding layer and all BERT layers.

While training this model for NLI task, we utilize the same sentence pair classification formulation used in the single-task BERT classifier. Figure 3.4 denotes the token embeddings used in both models for an example sentence pair. Moreover, the processing of the input is also the same as the baseline model for the NLI task. While processing an NLI input, we take the [CLS] token embedding at the BERT's output and treat it as the summary of the whole sequence. The dimension of this embedding is reduced to three after passing through a two-layer MLP since there are three labels in MultiNLI and SNLI. Finally, a softmax is applied to get the probability for each label class. In short, this model behaves just like the single-task BERT classifier while an NLI instance is being processed.

For the SRL training, we adapt an architecture similar to the one proposed by [Shi and Lin, 2019]. In our implementation, we indicate the predicate using the segment embeddings by assigning 1 to the predicate word pieces and 0 to the other tokens. Figure 3.6 denotes the embeddings used for the SRL. We opt for a simple decoder and rely purely on the self attention to capture contextual information. The embedding of each token is directly passed through a two-layer MLP independently, and the SRL tag is determined by a final softmax layer. We use the Inside Outside

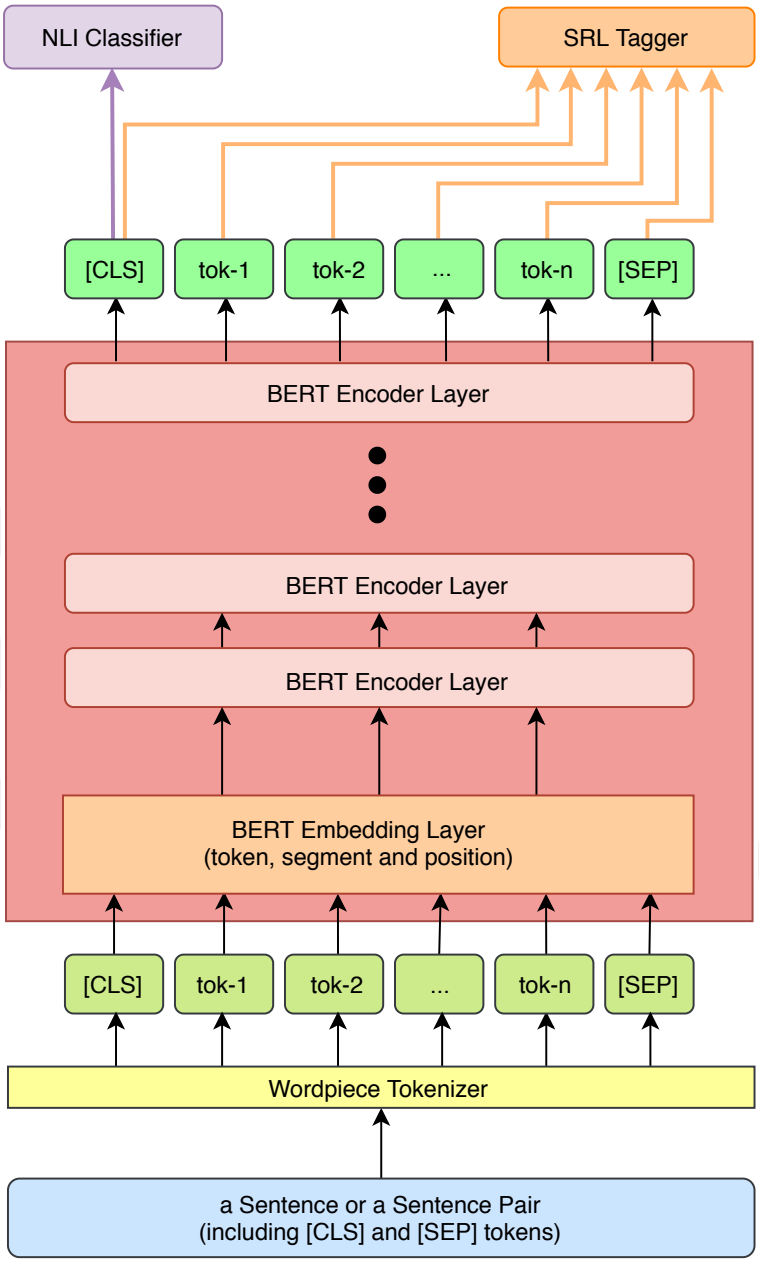


Figure 3.5: Multi-task BERT model for SRL and NLI. Each task-specific head contains a linear layer to transform the embeddings into the task's label space.

Beginning (IOB)[1] tagging for spans, and a Viterbi decoder to ensure prediction of valid spans during testing. As Figure 3.4 and 3.6 show, the segment embeddings

[1]In IOB style tagging, each span begins with a *B-* tag and continues with *I-* tags except for *Other* tokens, which takes *O* tags.

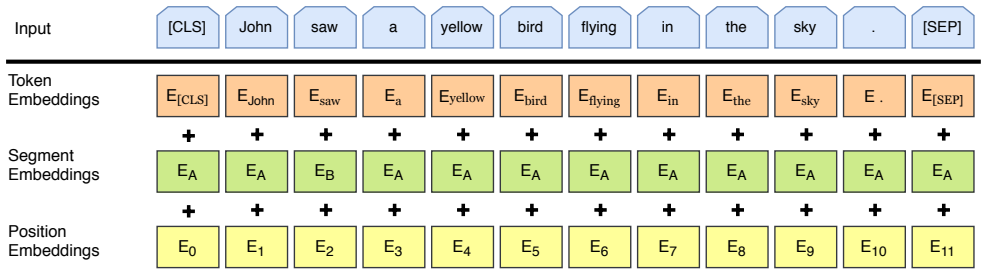


Figure 3.6: Input representation of *"John saw a yellow bird flying in the sky."* sentence for SRL task when the predicate is *saw*.

represent different things for SRL and NLI inputs. In NLI, the segment embeddings separate the sentences whereas they indicate the target verb in SRL. Moreover, how the BERT outputs are processed is also different between the tasks. In spite of these differences, the training is expected to optimize the BERT weights so that it can generate embeddings suitable for both tasks. Intuitively, this representation will be less prone to dataset biases in the NLI thanks to explicitly forcing the model to pay attention to the semantic roles of the words.

3.4 Binary Label Conversion

Popular NLI datasets, including MultiNLI and SNLI, generally have three labels which are the *entailment*, *contradiction* and *neutral*. However, the HANS dataset has binary labels i.e. the *entailment* and *non-entailment*. Therefore, we need to merge the *neutral* and *contradiction* labels into a single negative label, *non-entailment*, to make the model predictions compatible with HANS. There are two straightforward approaches to do that. The first one is doing the conversion on the training dataset, i.e. converting the MultiNLI dataset to binary by merging the labels as explained, and training the model on the resulting binary dataset. However, to apply this option, we need to change the size of the model's classification layer from three to two. The other option is to not change the model or dataset at all, and only change the model predictions as a post-processing step. Following [McCoy et al., 2019b], we take the latter approach. When we evaluate the model on HANS, we collapse

the predicted *contradiction* and *neutral* labels into a single negative *non-entailment* label to output binary labels.

Similarly, Comparisons dataset has two gold labels, *entailment* and *contradiction*. [Dasgupta et al., 2018] used Comparisons directly to evaluate the models that were trained on a three-class SNLI dataset. However, the drawback of this approach is that the *neutral* predictions will always be wrong since that class is not represented in this evaluation set. Therefore, we converted the predicted *neutral* labels into *contradiction* to unify the negative labels similarly to the treatment of HANS.

Chapter 4

EXPERIMENTS AND RESULTS

In this chapter, we present the experiments we conducted by training the BERT based models in single-task and multi-task learning setups [Cengiz and Yuret, 2020]. MultiNLI and SNLI datasets were used to train the models whereas HANS and Comparisons were used for evaluation. For the multi-task learning experiments, we used the SRL dataset for joint training with NLI. In both of the HANS and Comparisons experiments, we tuned the hyperparameters using a validation set from the same distribution as the adversarial test set. Nevertheless, we also tested our highest performing models on the original MultiNLI and SNLI validation sets to make sure our multi-task BERT model performs well on those as well. Indeed, the multi-task model performance on the original validation sets turned out to be similar to the single-task BERT performance (accuracy difference is around $\pm 0.5\%$, depending on the hyperparameters).

4.1 HANS Experiments

We experimented with direct training on NLI, sequential transfer learning using SRL, and a multi-task learning approach to train on the NLI and SRL tasks jointly. Since HANS was proposed as an adversarial evaluation set for the NLI models trained on MultiNLI, we use it as the NLI training dataset. Following [McCoy et al., 2019b], we output *non-entailment* label when a model predicts *contradiction* or *neutral*.

4.1.1 Single-task MultiNLI Training

We trained a single-task BERT model on the MultiNLI dataset to be used as the baseline for the HANS evaluation. The BERT weights are initialized with the pre-trained weights from [Devlin et al., 2018] whereas the classifier head is randomly

initialized. All weights are updated during training. We used the HANS augmentation dataset as the development set for hyperparameter tuning.

As reported by [McCoy et al., 2019b], BERT performs poorly on HANS although better than bag-of-words or LSTM [Hochreiter and Schmidhuber, 1997] based models. However, the follow-up work by [McCoy et al., 2019a] showed BERT's performance on HANS varied dramatically depending on the order of instances fed during training and the initial weights of the classifier head, both of which can be varied by changing the random seeds. To further investigate that, they repeated the training of BERT 100 times with the same settings except that randomness and compared the results. The largest variance was encountered on the *lexical overlap* category when the gold label is *non-entailment* whereas the other results were close among different runs. In our experiment, we got a 51% accuracy on *lexical overlap*, which is close to the upper limit of the range $(6\% - 54\%)$ reported by [McCoy et al., 2019a]. Table 4.1 includes the comparison of the original results [McCoy et al., 2019b] with our run (the single-task row). The results are comparable, so we use our result as the baseline for single-task BERT performance.

	Correct: Entailment			**Correct: Non-entailment**			
BERT model	**Lexical**	**Subseq.**	**Const.**	**Lexical**	**Subseq.**	**Const.**	**Avg.**
[McCoy et al., 2019b]	0.95	0.99	**1.00**	0.16	0.04	0.16	0.55
Single-task	**0.96**	**1.00**	0.99	0.51	0.05	0.18	0.62
Multi-task	0.91	0.98	0.95	**0.71**	**0.13**	**0.25**	**0.66**

Table 4.1: Comparison of the previous work [McCoy et al., 2019b], our single-task and our multi-task BERT models on HANS. All models started from pre-trained weights. The multi-task model was jointly trained on SRL and MultiNLI whereas the single-task models were trained only on MultiNLI. The highest accuracy for each category is indicated with bold. Note that [McCoy et al., 2019b] results on the *entailment* and *non-entailment* categories were obtained by averaging the subcategories using the BERT column of Table 7 and Table 8 in their paper respectively. (**Lexical**: lexical overlap, **Subseq.**: subsequence, **Const.**: constituent)

4.1.2 Multi-task Training for SRL and MultiNLI

In this part, we present the result of training BERT on SRL and MultiNLI jointly with the multi-task approach described in Section 3.3. We used the HANS augmentation dataset as the validation set for MultiNLI, and CoNLL-2012 development set for SRL validation. We validated the trained model against both validation sets separately at the end of each epoch. Then, the model performed highest on the HANS augmentation set was evaluated on the HANS test set. The results are shown in Table 4.1, together with the single-task results for comparison. The multi-task approach improved the overall accuracy by 4%. Although there is a slight decrease in the results when the correct label is *entailment*, this is an expected drop. The accuracy of the single-task model reaches 100% on the *subsequence* category, and is at least 96% on the remaining two categories when the correct label is *entailment*. Since the MultiNLI instances involving a heuristic examined by HANS are mostly labeled with *entailment*, the models tend to assign *entailment* labels to such examples in the HANS dataset. Because our multi-task model is less severely affected by the heuristics, it is less likely to output *entailment* when these heuristics are encountered. As one would expect, the gains come from the instances whose correct label is *non-entailment*. Noticeably, the multi-task training improved the accuracy for this label in all three categories dramatically. To examine the improvements in more detail, we present the results broken down into subcategories in Table 4.2. We refer the readers to [McCoy et al., 2019b] for the details and examples of the subcategories.

The top part of the Table 4.2 shows the fine-grained results for the *lexical overlap* category with the *non-entailment* gold labels. Although all subcategories improved, the largest gain comes from the *passive* examples. The *passive* case with *non-entailment* labels includes examples with sentence pairs almost identical to each other, only one of them has an active verb while the other has the passive form of it. An example sentence pair is the following: *"The senators were helped by the managers. ⇒ The senators helped the managers."* The single-task model misclassified almost all *non-entailment* examples involving passive sentences whereas the

Category	**Single-Task**	**Multi-Task**
Lexical Overlap		
Subject-object swap	0.68	0.83
Preposition	0.68	0.79
Relative clause	0.60	0.73
Conjunction	0.55	0.70
Passive	0.01	0.51
Subsequence		
NP/S ambiguity	0.00	0.03
Prepositional phrase on subject	0.14	0.20
Relative clause on subject	0.10	0.24
Past participle	0.00	0.06
NP/Z ambiguity	0.02	0.15
Constituent		
Embedded under if	0.46	0.71
After if clause	0.00	0.00
Embedded under verb	0.31	0.47
Disjunction	0.07	0.02
Adverb	0.08	0.06

Table 4.2: Fine-grained comparison of single-task and multi-task BERT on HANS's three different categories when the correct label is *non-entailment*. The rows denote the heuristic type found in the sentence pairs.

multi-task model could predict them correctly half of the time. This is a significant improvement, the multi-task model has begun to identify the meaning changes when a verb is switched from passive to active while the word order is kept unchanged.

The middle section of Table 4.2 shows the results on the *subsequence* category for the *non-entailment* gold labels. All subcategories improved although the degree is less than *lexical overlap*. The largest improvement is found on *relative clause on subject* which represents the sentence pairs differing by their subjects such that the premise's subject is a relative clause and the hypothesis's subject is a particular segment of that clause that leads to *non-entailment*. An example sentence pair from this category is the following: *"The secretary that admired the senator saw the actor. ⇒ The senator saw the actor."*. When trained with a multi-task approach, the model makes some progress on recognizing that the overall meaning of a relative clause does not necessarily entail a part of it.

The last category we investigated is *constituent*s with *non-entailment* gold labels whose results are given in the bottom part of Table 4.2. There are significant improvements in two subcategories whereas the remaining three did not improve or very slightly dropped. The largest improvement (25% accuracy) is on *embedded under if* subcategory which denotes the examples with a premise having an *if* (or *unless*) clause whereas the hypothesis has the result part of the *if* clause. An example from this subcategory is *"Unless the authors saw the students, the doctors resigned. ⇒ The doctors resigned."* The second largest gain (16% accuracy) comes from *embedded under verb* subcategory which is similar to the previous one, except that the embedding is achieved using a verb. An example is *"The tourists said that the lawyer saw the banker. ⇒ The lawyer saw the banker."*.

As shown by the HANS results, there are solid improvements on NLI evaluation after switching to multi-task training. However one needs to ask if the joint training hurts the other task, SRL. In the multi-task experiment, the F1 score on the SRL test set is 86.0 which is comparable to the single-model SOTA result noted by [Shi and Lin, 2019]. Therefore, the joint training did not harm the SRL performance, while improving the out-of-distribution performance on NLI.

4.1.3 Sequential Transfer Learning from SRL to MultiNLI

We experimented with sequential transfer learning to test if a simple transfer learning strategy is enough to carry information from the SRL task so that the model is more robust to the biases in the NLI dataset. First, an SRL tagger head with random weights is appended on top of the pre-trained BERT encoder. This model is fine-tuned on SRL until the F1 score on the SRL validation set is maximized. The model weights from the epoch resulting in the highest SRL development set score is stored. Then, its SRL head is stripped, and an NLI classifier head with random weights is appended on top of the [CLS] token. Finally, the model is trained on MultiNLI and validated against HANS augmentation set. After training, the model is evaluated on the HANS test set. The result is within the accuracy range for the single-task training results reported by [McCoy et al., 2019a]. This shows that our transfer learning strategy did not improve HANS results over the BERT trained only on MultiNLI. We anticipate that this is because the model forgets most of the knowledge about the SRL task during NLI training. To avoid that, we switched to multi-task setup presented in Section 3.3 to learn SRL and NLI jointly so that the semantic role knowledge is not forgotten.

4.2 Comparisons Experiments

We trained BERT with the single-task and multi-task learning approaches and compared them on the Comparisons dataset. First, we used the SNLI dataset as the NLI training source following [Dasgupta et al., 2018]. We used the validation set released with the Comparisons for hyperparameter optimization during training of both the single-task and multi-task models. Unlike SNLI, this dataset contains only two labels, *entailment* and *contradiction*. Therefore, differently than [Dasgupta et al., 2018], we converted the predicted *neutral* labels to *contradiction* to have a unified negative label. Table 4 compares the overall performance of our BERT based models and the previously examined models on the test set. InferSent [Conneau et al., 2017a] is a sentence encoding based NLI model that uses LSTM as the encoder. Although it is more complex than the bag-of-words (BOW-MLP) model, their per-

Model	**same**	**more /less**	**not**	**Avg.**
BOW-MLP	50.0	50.0	49.9	50.0
InferSent	51.4	**50.1**	47.8	49.8
BERT	**85.3**	47.9	44.5	59.2
MTL-BERT	80.5	47.9	**51.3**	**59.9**

Table 4: Percent accuracy of the models on Comparisons dataset after training on SNLI. BERT based models are our implementations while the others are from [Dasgupta et al., 2018]. Multi-task (MTL) BERT is trained on SRL and SNLI. The highest accuracy for each category is indicated with bold. Note that the BOW-MLP and InferSent rows are obtained by merging the *neutral* and *contradiction* labels in Figure 2 and 3 from [Dasgupta et al., 2018].

formances are similar on this dataset. We see that the performance of BERT models on the *same* category are much better than the simpler models. The high performance of BERT models on this category can be attributed to the fact that BERT was pre-trained on a large corpus with missing word prediction and next sentence prediction tasks, making it more aware of the word order. However, in the remaining two categories, both the single-task and multi-task BERT perform relatively close to the remaining models.

MultiNLI is a more diverse dataset compared to SNLI, including examples from several genres. Therefore, we also tried MultiNLI as the NLI training source and replicated the Comparisons experiments to see how the single-task and multi-task BERT performance will change. The results are given in Table 5 together with the SNLI based results for comparison. We see that switching to MultiNLI improved both models substantially. However, the increase in the multi-task model is significantly more prominent, showing the advantage of the joint training with SRL. The multi-task model correctly classifies almost all test examples in the *more/less* category and most of the *not* category. However, the trend of observing better performance on the *same* category from the single-task model holds here as well. This

	Training set: SNLI				Training set: MultiNLI			
BERT Model	**same**	**more/less**	**not**	**Avg.**	**same**	**more/less**	**not**	**Avg.**
Single-task	85.3	47.9	44.5	59.2	74.1	88.3	74.3	78.9
Multi-task	80.5	47.9	51.3	59.9	63.3	97.3	91.9	84.2

Table 5: Percent accuracy of the BERT models on Comparisons dataset.

result is surprising and needs further investigation.

4.3 Training Details

We used the PyTorch [Paszke et al., 2017] framework and the AllenNLP [Gardner et al., 2018] library for implementation. We adapted some code to implement the multi-task training logic from [Sanh et al., 2019]'s hierarchical multi-task learning project[1]. In all experiments, we used the base version of BERT by initializing it with the weights released by [Devlin et al., 2018].

We use uniform mini-batches, i.e. a mini-batch contains instances from a single-task. Each dataset is divided into mini-batches with the same size and an iterator for each of them is created that can cycle through a dataset and provide batches indefinitely. In a training step, we decide which task to train with a probabilistic sampling, get a mini-batch from the iterator for that task, and perform a forward pass on it and back-propagate the loss. During the back-propagation, we update the task-specific head of the chosen task, as well as the BERT encoder. Following [Sanh et al., 2019], we use proportional sampling to decide on the task type at the beginning of each training step.

Recent studies generally use a single global optimizer for all tasks [Sanh et al., 2019, Liu et al., 2019]. In this work, we tried both this approach and using a different optimizer for each task. The advantage of using multiple optimizers is that the learning rates of the individual tasks can be set to different values, and each task can have its own learning rate scheduler. We used *BertAdam* optimizer from

[1] `https://github.com/huggingface/hmtl`

Hugging Face, and set its maximum learning rate to 2e-5 or 5e-5 according to the validation accuracy on the NLI evaluation task. Moreover, we employed a slanted triangular learning rate scheduler [Howard and Ruder, 2018] with a cut fraction of 0.1 and decay factor of 0.38. In all experiments the maximum sequence length is set to 256, and longer sequences are truncated. In all training experiments, 4 GPUs were used in parallel and the datasets were divided into mini-batches of size 12 based on GPU memory limitations.

Chapter 5

RELATED WORK

In this chapter, we discuss various solution approaches proposed for the NLI task. We start with a sentence embedding based approach that has been successfully applied to various NLI tasks. Then, we present a transfer learning approach from our previous work on NLI in a special domain with limited data. We conclude the discussion with approaches targeting the generalization problem of NLI models. These include supporting the models with syntax or semantic roles and multi-task learning methods.

5.1 Sentence Embedding Models

Some previous studies used the sentence embedding based approaches to solve NLI. Noticeably, InferSent [Conneau et al., 2017a] uses an LSTM to encode the premise

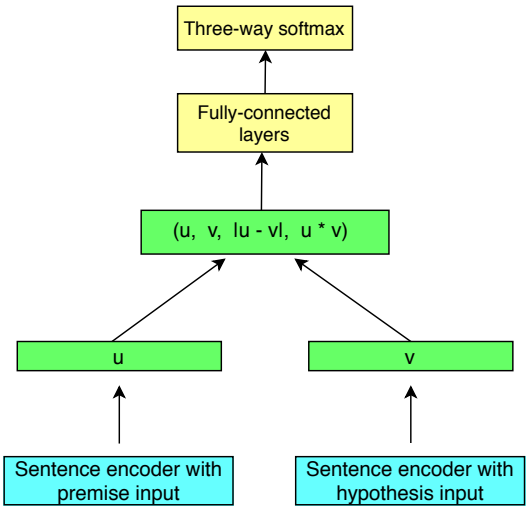


Figure 5.1: InferSent architecture from [Conneau et al., 2017a]. The sentence encoder is an LSTM.

and hypothesis sentences independently. Then, it concatenates the premise, hypothesis embeddings and two feature vectors obtained by their element-wise multiplication and absolute difference to get the overall sentence pair representation. Finally, an MLP layer followed by a softmax is used to calculate the class scores. Figure 5.1 depicts the overall architecture. Being trained on SNLI, this model suffers from the biases in its training data and performs close to the BOW model on the Comparisons evaluation set.

5.2 Transfer Learning for Training on Low-resource Domain

In our previous work [Cengiz et al., 2019], we showed that employing transfer learning is very useful to train BERT on an NLI dataset from a low-resource domain. That dataset, MedNLI [Romanov and Shivade, 2018], was constructed from clinical notes about patients, written by healthcare professionals. The training set is small (11232 instance pairs), so training a model on it directly is likely to result in overfitting. Moreover, the dataset was made of medical texts and contains medical jargon. That makes it harder for the word embedding based models since the word embeddings are typically trained on open-domain datasets written in plain English. To address this problem, we started with BERT, trained it on large NLI datasets such as MultiNLI [Williams et al., 2018] and SNLI [Bowman et al., 2015] to support the inference reasoning, and then trained it further on our target task, MedNLI. In the first training stage, we transfer the knowledge of the task, NLI, into our pre-trained model. During the second training stage, we specialize our model on the MedNLI dataset. We showed that this approach produces higher accuracy on MedNLI compared to direct application of BERT.

5.3 Data Augmentation to Combat the Heuristics

There are a number of studies that use data augmentation to address the generalization problem revealed by NLI challenge datasets like HANS and Comparisons. Noticeably, [McCoy et al., 2019b, Dasgupta et al., 2018, Nie et al., 2018] created augmentation sets consisting of training instances with properties similar to the pro-

posed adversarial evaluation set. The augmentation set is focused on the examined phenomena and considerably smaller than the original training set in general. Nevertheless, the models are shown to achieve very strong results, even close to %100 accuracy on the evaluation sets after training with augmented datasets. However, there are some problems with an augmentation approach performed this way, i.e. using a new dataset targeting the inspected phenomena in the proposed evaluation set. First of all, it is not clear if they do result in improvement on the language understanding of the model in general. Rather, the model at hand is patched so that it can excel on some specific cases that the new evaluation dataset examines. However, one can presumably find other adversarial example classes for a given training dataset, thus creating an augmentation set for each possible adversarial class may not be feasible. Moreover, [Nie et al., 2018] showed that augmenting the training dataset by targeting some specific category of adversarial examples might be harmful to other types of adversarial examples. In other words, dataset augmentation with such limited focus might lead to overfitting to the targeted adversaries and hurt the overall robustness. Therefore, we took a different approach in this work, and introduced an inductive bias on the model by explicitly enforcing it to produce representations suitable to extract semantic role information.

5.4 Using Low-level Information to Improve the NLI Performance

Some recent studies have investigated the benefits of semantic role labeling on the performance of natural language inference models. Noticeably, [Zhang et al., 2020, Zhang et al., 2018] used SRL as a supplementary task for text comprehension tasks such as textual entailment and question answering. Similar to our work, they used PropBank [Palmer et al., 2005] style role annotations and treated SRL as a sequence tagging problem. [Zhang et al., 2020]'s approach is different from ours in that they use a pre-trained, SOTA SRL model to provide semantic embeddings to enrich the contextual embeddings from BERT. They kept the SRL model frozen, but trained other parts of the model including the BERT encoder. Similarly, [Zhang et al., 2018] employed two different networks, one of which is an SRL model responsible for

generating the semantic embeddings to support the other network which is trained to solve the downstream task at hand. Moreover, both networks in this model use pre-trained word embeddings such as ELMo [Peters et al., 2018] or GloVe [Pennington et al., 2014]. The main difference of our approach from those is that we use a single network, and train it in a multi-task fashion by sharing the encoder representations among different tasks. Moreover, unlike our work, they evaluated their models on the original datasets e.g. MultiNLI, so their focus was not to improve the model performance on the adversarial evaluation sets.

Instead of semantic role information, some recent works investigated the benefits of syntax to support the natural language inference models. Noticeably, [Pang et al., 2019] used the hidden word representations of an externally trained, high performing dependency parser to enrich the BERT based NLI models. With this approach, the models achieved some modest increase on the overall HANS results.

5.5 Multi-task Learning Methods

Multi-task models powered with pre-trained language model based encoders have achieved SOTA performance on natural language understanding benchmarks such as GLUE [Wang et al., 2018] and SuperGLUE [Wang et al., 2019]. However, there is not much work focusing on simultaneously solving both a word-level semantics task such as SRL and a sentence-level understanding task such as NLI which requires higher level reasoning. Instead, the existing approaches such as [Liu et al., 2019, Clark et al., 2019] combine multiple sentence-level understanding tasks to solve them jointly without any aid from lower level tasks focusing on syntax or word-level semantics. Our approach differs from them in that we hypothesize using both word-level semantics and high level reasoning tasks might be a more suitable approach to learn deeper understanding of the sentences, thereby suffering less from the dataset biases in reasoning tasks such as NLI.

Some previous work attempted to cast NLI to a different natural language understanding task such as question answering. Particularly, [McCann et al., 2018] suggested a collection of various tasks including NLI as a benchmark and proposed

a novel approach to solve all those tasks using a single multi-task model. They casted each task to the question answering problem and trained a model to solve all of them jointly.

Chapter 6

CONCLUSION AND FUTURE WORK

This thesis presented a multi-task learning approach using SRL to apply an inductive bias on a BERT based NLI model. Our experiments show that joint training with SRL makes the model more robust to the superficial patterns in the NLI training data. As opposed to the augmentation based solutions focused on specific adversarial classes, this approach has the advantage of being applicable to a variety of adversaries without overfitting to some of them. Having access to the semantic role information improves the sentence understanding of the model, hence making it generalize better to the unseen dataset distributions including the adversarial ones such as HANS and Comparisons. Moreover, our experiments show that the proposed multi-task learning approach is more effective than the sequential transfer learning on HANS.

The SRL task utilized in this work processes a single predicate per data instance. However, considering all predicates simultaneously would probably be more helpful to improve the model's understanding of the sentences. Therefore, the future work might incorporate joint prediction of all predicates and corresponding roles to analyze its effect on adversarial NLI evaluation performance.

BIBLIOGRAPHY

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[Cengiz et al., 2019] Cengiz, C., Sert, U., and Yuret, D. (2019). KU_ai at MEDIQA 2019: Domain-specific pre-training and transfer learning for medical NLI. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 427–436, Florence, Italy. Association for Computational Linguistics.

[Cengiz and Yuret, 2020] Cengiz, C. and Yuret, D. (2020). Joint training with semantic role labeling for better generalization in natural language inference. In *Proceedings of the 5th Workshop on Representation Learning for NLP (RepL4NLP-2020)*, Seattle, WA, USA. Association for Computational Linguistics. (to appear).

[Clark et al., 2019] Clark, K., Luong, M.-T., Khandelwal, U., Manning, C. D., and Le, Q. V. (2019). BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

[Conneau et al., 2017a] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017a). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

[Conneau et al., 2017b] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017b). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[Dasgupta et al., 2018] Dasgupta, I., Guo, D., Stuhlmüller, A., Gershman, S. J., and Goodman, N. D. (2018). Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Computing Research Repository*, arXiv:1810.04805.

[Gardner et al., 2018] Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

[He et al., 2017] He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

[Kingsbury and Palmer, 2002] Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

[Lakew et al., 2018] Lakew, S. M., Cettolo, M., and Federico, M. (2018). A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 641–652, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[Liu et al., 2019] Liu, X., He, P., Chen, W., and Gao, J. (2019). Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

[MacCartney, 2009] MacCartney, B. (2009). *Natural Language Inference*. PhD thesis, Stanford University, Stanford, CA, USA. AAI3364139.

[McCann et al., 2018] McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

[McCoy et al., 2019a] McCoy, R. T., Min, J., and Linzen, T. (2019a). Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance.

[McCoy et al., 2019b] McCoy, T., Pavlick, E., and Linzen, T. (2019b). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

[Nie et al., 2018] Nie, Y., Wang, Y., and Bansal, M. (2018). Analyzing compositionality-sensitivity of NLI models. *CoRR*, abs/1811.07033.

[Palmer et al., 2005] Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

[Pang et al., 2019] Pang, D., Lin, L. H., and Smith, N. A. (2019). Improving natural language inference with a pretrained parser.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

[Pradhan et al., 2013] Pradhan, S., Moschitti, A., Xue, N., Ng, H. T., Björkelund, A., Uryupina, O., Zhang, Y., and Zhong, Z. (2013). Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

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

[Romanov and Shivade, 2018] Romanov, A. and Shivade, C. (2018). Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

[Sanh et al., 2019] Sanh, V., Wolf, T., and Ruder, S. (2019). A hierarchical multi-task approach for learning embeddings from semantic tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6949–6956.

[Shi and Lin, 2019] Shi, P. and Lin, J. (2019). Simple bert models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[Wang et al., 2019] Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

[Wang et al., 2018] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

[Williams et al., 2018] Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., John-

son, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository*, arXiv:1609.08144.

[Zhang et al., 2018] Zhang, Z., Wu, Y., Li, Z., and Zhao, H. (2018). Explicit contextual semantics for text comprehension.

[Zhang et al., 2020] Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., and Zhou, X. (2020). Semantics-aware BERT for language understanding. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2020)*.