

# Structure and Feedback-based Natural Language Processing

by

Dhanasekar Sundararaman

Department of Electrical and Computer Engineering  
Duke University

Date: \_\_\_\_\_

Approved:

---

Lawrence Carin, Supervisor

---

Hai Li

---

Ricardo Henao

---

Robert Calderbank

---

Rabih Younes

Dissertation submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy in the Department of Electrical and Computer Engineering  
in the Graduate School of Duke University

2022

ABSTRACT

**Structure and Feedback-based Natural Language  
Processing**

by

Dhanasekar Sundararaman

Department of Electrical and Computer Engineering  
Duke University

Date: \_\_\_\_\_

Approved:

\_\_\_\_\_  
Lawrence Carin, Supervisor

\_\_\_\_\_  
Hai Li

\_\_\_\_\_  
Ricardo Henao

\_\_\_\_\_  
Robert Calderbank

\_\_\_\_\_  
Rabih Younes

An abstract of a dissertation submitted in partial fulfillment of the  
requirements for the degree of Doctor of Philosophy  
in the Department of Electrical and Computer Engineering  
in the Graduate School of Duke University  
2022



# Abstract

The development of deep learning models has revolutionized the way information is processed by computers and has made significant advancements in fields such as speech, vision, and language. In language, giant strides have been made ranging from seq2seq models that process one word at a time to more sophisticated Transformer networks that can feed on paragraphs of text. Due to their ability to generate coherent and meaningful sentences at scale, natural language processing (NLP) models have become so prevalent. Despite their effectiveness, these models often have room for improvement when presented with additional linguistic information. In this dissertation, I discuss my contributions to the use of (a) structural information, namely syntax, and numeric structures, often overlooked and underutilized by language models, and (b) feedback-based models in which the objectives of the main model are guided by feedback from a supporting model.

In the first part, I will present three of my contributions which explore the use of structural information to develop effective NLP models that outperform their baselines on tasks that require encoders and decoders, such as machine translation, as well as downstream tasks, such as text classification, question answering, fill-in-the-blanks, etc. The first contribution proposes techniques for consuming syntactic information such as part of speech, word position, and case in order to improve the performance of machine translation on data-heavy Transformer models. An accompanying case study compares and contrasts a seq2seq model with a Transformer in its ability to absorb syntax across many language pairs. The second and third contributions concern utilizing a numeric structure that is prevalent in languages, as a means of incorporating numeral reasoning into language models. Collectively, these contributions contribute to the improvement of translation performance, numerical question-answer reasoning, and other downstream tasks.

In the second part, I will present my two contributions that utilize feedback signals from a supporting model to achieve an optimization objective that enhances the performance of the main model. The first contribution deals with the main model as a multi-task model that performs language inference across multiple task languages, whereas the supporting model uses reinforcement learning to ascertain the importance of each task that is not known apriori. Based on this approach, the resultant mix of tasks led to significant improvements in the performance of the target language task. In the second contribution, a supporting model is used to select tokens that will most likely be out-of-distribution (OOD) tokens by using Mahalanobis distance and performing a technique known to language models as self-supervision. Using a novel regularization loss, the distance between in-domain tokens and pseudo-OOD tokens is maximized, which results in significant performance improvements when detecting OODs.

# Contents

<b>Abstract</b>	<b>iv</b>
<b>List of Tables</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiii</b>
<b>Acknowledgements</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Natural Language Processing . . . . .	1
1.2 Structure-based NLP . . . . .	4
1.3 Feedback-based NLP . . . . .	5
<b>2 Syntax-infused Transformers and BERT</b>	<b>8</b>
2.1 Introduction . . . . .	9
2.2 Related Work . . . . .	11
2.3 Background . . . . .	12
2.3.1 Baseline Transformer . . . . .	12
2.3.2 BERT . . . . .	13
2.4 Knowledge-Infused Models . . . . .	14
2.4.1 Syntax-infused Transformer . . . . .	14
2.4.2 Syntax-infused BERT . . . . .	16

2.5	Datasets and Experimental Details . . . . .	17
2.5.1	Machine translation . . . . .	18
2.5.2	Natural language understanding . . . . .	19
2.6	Experimental Results . . . . .	19
2.6.1	Machine translation . . . . .	19
2.6.2	Natural language understanding . . . . .	22
2.7	Conclusions . . . . .	23
<b>3</b>	<b>Syntax and Lexical Semantics</b>	<b>27</b>
3.1	Introduction . . . . .	28
3.2	Methods . . . . .	29
3.2.1	Models . . . . .	29
3.2.2	Dataset . . . . .	30
3.2.3	Word order and lexical similarity . . . . .	31
3.2.4	Training details . . . . .	32
3.3	Results and Discussion . . . . .	32
3.4	Conclusion . . . . .	34
<b>4</b>	<b>Numeracy-preserving Word Embeddings</b>	<b>36</b>
4.1	Introduction . . . . .	37
4.2	Related Work . . . . .	39
4.3	Methods . . . . .	40
4.3.1	DICE embeddings . . . . .	41
4.4	Experiments . . . . .	43
4.4.1	Task 1: Exploring Numeracy . . . . .	44

4.4.2	Task 2: List Maximum, Decoding, and Addition . . . . .	46
4.5	Applications of DICE . . . . .	49
4.5.1	Magnitude Classification . . . . .	49
4.5.2	Model-Based Numeracy Embeddings . . . . .	51
4.6	Training details for Magnitude Classification Experiment . . . . .	54
4.7	Hyperparameter for BERT + $\mathcal{L}_{num}$ . . . . .	54
4.8	Examples for BERT vs. BERT + $\mathcal{L}_{num}$ . . . . .	55
4.9	Conclusion . . . . .	56
<b>5</b>	<b>Number Entity Recognition</b>	<b>57</b>
5.1	Introduction . . . . .	57
5.2	Related Work . . . . .	60
5.3	Preliminaries . . . . .	61
5.4	Methods . . . . .	61
5.4.1	Number entity recognition . . . . .	61
5.4.2	Jointly trained embeddings . . . . .	62
5.4.3	FITB . . . . .	64
5.5	Results and Discussion . . . . .	64
5.5.1	Number entity recognition . . . . .	64
5.5.2	Jointly trained embeddings . . . . .	66
5.5.3	FITB . . . . .	67
5.6	Conclusions . . . . .	68
<b>6</b>	<b>Learning Task Sampling Policy for Multitask Learning</b>	<b>69</b>
6.1	Introduction . . . . .	69

6.2	Related Works . . . . .	71
6.3	Methods . . . . .	72
6.3.1	Learning Task Sampling Policy . . . . .	72
6.3.2	Exploration Strategy . . . . .	75
6.4	Experiments . . . . .	76
6.4.1	Training Settings . . . . .	77
6.4.2	Experimental Results . . . . .	77
6.5	Conclusions . . . . .	78
<b>7</b>	<b>Masking-based OOD Detection</b>	<b>79</b>
7.1	Introduction . . . . .	80
7.2	Related Works . . . . .	81
7.3	Preliminaries and Notations . . . . .	81
7.4	Post hoc Pseudo-OOD Regularization . . . . .	82
7.4.1	Masking for Pseudo-OOD Generation . . . . .	83
7.4.2	Pseudo-OOD Regularization . . . . .	85
7.5	Experiments . . . . .	85
7.5.1	Datasets . . . . .	87
7.5.2	Experimental Setup . . . . .	87
7.5.3	Results . . . . .	88
7.6	Tasks . . . . .	88
7.7	OOD Evaluation Metrics . . . . .	89
7.8	Adaptation of MASKER Baseline . . . . .	89
7.9	Ablation for choosing keywords . . . . .	90

7.10 Conclusions . . . . .	90
<b>8 Conclusions</b>	<b>91</b>
<b>Bibliography</b>	<b>92</b>
<b>Biography</b>	<b>109</b>

# List of Tables

2.1	BLEU scores for different proportions of the data for baseline Transformer vs syntax-infused Transformer for the EN-DE task on newstest2014. . . . .	17
2.2	BLEU scores improvements on low resource languages including Sinhala and Korean for the baseline and syntax-infused models. . . . .	18
2.3	Translation examples of baseline Transformer vs. syntax-infused Transformer on the EN-DE dataset. . . . .	20
2.4	GLUE test results scored using the GLUE evaluation server. The number below each task denotes the number of training examples. . . . .	21
2.5	Examples of randomly chosen sentences from the RTE dataset that were classified by BERT <sub>BASE</sub> + POS but not by BERT <sub>BASE</sub> . . . . .	21
3.1	Translation performance on target languages from OpenSubtitles2016 database	30
4.1	Performance (% accuracy) on numeracy tests. . . . .	43
4.2	Experimental results on list maximum, decoding, and addition using the DICE- <i>D</i> method. . . . .	44
4.3	Performance (%) on classifying number magnitude on the Numeracy-600k dataset. . . . .	50
4.4	Performance (%) of BiLSTM-attention with DICE model on the Numeracy-600k dataset by varying the embedding dimensions of input tokens. . . . .	50
4.5	F1 scores of BERT-base model on SQuAD 1.1 sub-splits. . . . .	54
5.1	Number of entities in our annotation for each type in SQuAD-Num. . . . .	60

5.2	Performance metrics for NuER on SQuAD-Num and ENT-Numeracy 600K for each of the six entity types. . . . .	63
5.3	Few-shot transfer vs fine-tuning performance on NuER entity types. . . . .	63
5.4	Improvements on SQuAD-Num dataset using Jointly trained Embedding technique with <i>BERT<sub>B</sub></i> (base), <i>BERT<sub>L</sub></i> (large), and <i>RoBERTa<sub>B</sub></i> (base). . . . .	65
5.5	Predictive performance on the FITB task. All numbers are tested for significance and have a standard deviation of 0.01 . . . . .	67
6.1	Performance improvements on the XNLI test dataset by our proposed methods. . . . .	75
6.2	Performance improvements on low-resources GLUE tasks, RTE, and MPRC, when using high-resource GLUE tasks. . . . .	76
7.1	AUROC and FPR@90 of Baseline and POORE on the three target benchmarks. . . . .	86
7.2	Masker baseline and Adapted approaches. . . . .	90
7.3	Ablation for keywords . . . . .	90

# List of Figures

1.1	Word2Vec model. . . . .	2
1.2	Recurrent Neural Networks . . . . .	2
1.3	Proposed DICE embeddings. . . . .	5
2.1	(a) Proposed formation of attention matrices. (b) The BERT <sub>BASE</sub> + POS model. . . . .	25
2.2	Comparison of attention for example sentences translated by baseline and POS Transformer models. . . . .	26
3.1	Performance improvements with POS models. . . . .	33
4.1	Proposed Deterministic, Independent-of-Corpus Embeddings (DICE). . . .	40
4.2	Qualitative examples where BERT + $\mathcal{L}_{num}$ performed better than BERT-base . . . . .	52
4.3	Qualitative examples where BERT + $\mathcal{L}_{num}$ performed better than BERT base. . . . .	55
5.1	Sequence classification of numbers . . . . .	58
5.2	Qualitative comparison of predictions of numbers on the FITB task between the BERT base and $BERT_{NuER}$ . . . . .	66
6.1	The sampling schedule learned for different target languages across meta-steps. . . . .	73
7.1	AUROC ( $\uparrow$ ) and FPR@90 ( $\downarrow$ ) using Maxprob, ODIN, Entropy, and BERT. . . . .	86

# Acknowledgements

My Ph.D. journey was filled with a great deal of learning, opportunities, achievements, as well as challenges. In the course of my Ph.D., I was fortunate to receive valuable support from some of the best minds in research from the Duke community and beyond. Here, I'd like to express my sincere gratitude to many people who have helped me along the way.

Firstly, I would like to express my sincere gratitude to my advisor, Dr. Lawrence Carin (Larry), who has provided me with ample freedom to pursue my research interests while also motivating and guiding me throughout the entire process. My motivation and drive for completion were primarily influenced by his profound knowledge, commitment to research, and principles. Aside from Larry's expertise, he is also an excellent leader, who led our group with impact and taught me the importance of interpersonal skills when working with others. His empathy made it possible to weather some difficult times, including the pandemic.

Next, I would like to thank the members of my dissertation committee, Dr. Hai Li, Dr. Ricardo Henao, Dr. Robert Calderbank, and Dr. Rabih Younes, for their support and valuable feedback on my thesis. Additionally, I gained valuable experiences from my interactions with them during the course of a variety of academic endeavors.

I'm also very grateful to the various mentors, hosts, and friends I made during my internships at Microsoft, Google, and Amazon. They helped me obtain an industry perspective, and also the real-world applications of my research.

I would also like to thank my fellow group members who I had discussed and collaborated with on many occasions. I want to thank: Vivek Subramanian, Guoyin Wang, Nikhil Mehta, Dinghan Shen, Pengyu Cheng, Shijing Si, Serge Assaad, Dong Wang, Chenyang Tao, Yulai Cong, Travis Maxfield, Shounak Dutta, Chunyuan Li, Wenlin Wang, Yitong

Li, Xinyuan Zhang, Kevin Liang, Gregory Spell, Liqun Chen, Shuyang Dai, Ruiyi Zhang, Paidamoyo Chapfuwa, Siyang Yuan, Hao Fu, Weituo Hao, Jiachang Liu, Rui Wang, Yuan Li, Bai Li, Jianyi Zhang, Hongteng Xu, and Hao Zhang.

Finally, I would like to thank my dear parents for their unconditional love and support. As a result of their support, I have come this far in life and I am truly fortunate to have them by my side.

# Introduction

## 1.1 Natural Language Processing

Natural language processing (NLP) is one of the most prominent fields in machine learning impacting day-to-day applications such as conversational assistants, translations, question answering, and applications of text generation. Word2Vec [MSC<sup>+</sup>13], a technique for generating distributional vectors or word embeddings, introduced the idea of converting raw texts into representational vectors. A word embedding is based on the distributional hypothesis, the idea that words appearing in similar contexts should have similar meanings. CBOW and skip-gram construct word embeddings of target words using surrounding context words and vice versa, respectively. Word2Vec has the substantial limitation of only approximating based on a limited number of words, often only a few.

$$F(\theta) = \sum_{t=1}^T \sum_{-m \leq j \leq m; j \neq 0} P(w_{t+j}/w_t; \theta) \quad (1.1)$$

where  $m$  represents the window in a Word2Vec model,  $F$  represents the objective

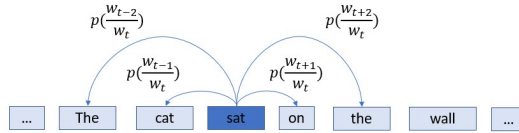


FIGURE 1.1: Word2Vec model.

function,  $T$  represents the total number of words and  $\theta$  the parameters.

A new breed of neural networks called recurrent neural networks (RNNs) was introduced that are designed to process sequences of words one at a time and are ideally suited to tasks such as machine translation. In the case of RNNs, there is a problem of vanishing gradients, making it difficult to approximate gradients in order to retain long sequences of information. As a solution to this problem, LSTM networks [HS97] offered additional parameters for remembering and forgetting parts of sequences. In nearly all of the NLP tasks, RNN networks powered by LSTM outperformed vanilla RNN networks. Although LSTM performed well in terms of its ability to retain information about distant tokens, it still had all the disadvantages associated with a vanilla RNN model. In RNN networks, scaling is challenging due to the fact that parallelization is almost impossible. The generation of any token is dependent upon all the tokens that have been generated before, and this chain has led to incredibly inefficient computations and time constraints. As a result, highly parallelizable attention networks [VSP<sup>+</sup>17] were developed in order to overcome these issues.

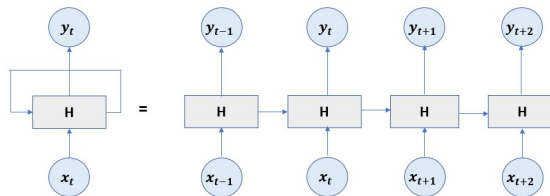


FIGURE 1.2: Recurrent Neural Networks

$$h_t = f_w(h_{t-1}, x_t) \tag{1.2}$$

where  $h_t$  represents the current state of a RNN dependent on the past state  $h_{t-1}$  and input  $x_t$ .

Attention-based deep learning models for natural language processing (NLP) have shown promise for various machine translation and natural language understanding tasks. For word-level sequence-to-sequence tasks such as translation, paraphrasing, and text summarization, attention-based models allow a single token (also called subwords) to be represented as a combination of all tokens in the sequence [LPM15]. The distributed context allows attention-based models to infer rich representations for tokens, leading to more robust performance. One such model is the Transformer, which features a multi-headed self and cross-attention mechanism that allows many different representations to be learned for a given token in parallel [VSP<sup>+</sup>17]. The encoder and decoder arms contain several identical stacked subunits to learn embeddings for tokens in the source and target vocabularies.

In spite of the fact that attention-based deep learning models have significantly transformed text generation and its applications, studies have shown that their performance can be further enhanced when complemented with additional linguistic information [SH16]. In this dissertation, I will discuss the usefulness of structural information and feedback-based models for natural language processing. Specifically, I examine the impact of infusion of syntactic and numerical structure on the NLP models and evaluate their performance in a variety of downstream tasks. In the case of feedback-based models, I examine the impact of feedback signals from a supporting model that could affect the performance of the main model. As part of this study, I explore a reinforcement learning framework for learning

task weights in a multi-task setup that could enhance performance on the target task. In addition, I examine the feedback signals obtained by means of Mahalanobis distance to identify potential pseudo-OOD tokens in order to improve OOD detection.

## 1.2 Structure-based NLP

Syntax is an essential feature of grammar from external knowledge that facilitates the generation of coherent sentences. For instance, part of speech (POS) dictates how words relate to one another (*e.g.*, verbs represent the actions of nouns, adjectives describe nouns, *etc.*). Studies show that when trained for a sufficiently large number of steps, NLP models can potentially learn underlying patterns about text like syntax and semantics, but this knowledge is imperfect [JSS<sup>+</sup>19]. However, works such as [KDH<sup>+</sup>18], [LDG16] show that NLP models that acquire even a weak understanding of syntactic structure through training demonstrate improved performance relative to their baseline counterparts. Hence, explicit prior knowledge of syntactic information can benefit NLP models in various tasks [SSW<sup>+</sup>19, SSW<sup>+</sup>21, SS21].

While word embeddings effectively capture semantic relationships between *words*, but they are less effective at capturing numeric properties associated with *numbers*. Though numbers represent a significant percentage of tokens in a corpus, they are often overlooked. In non-contextual word embedding models, they are treated like any other word, which leads to misinterpretation. For instance, they exhibit unintuitive similarities with other words and do not contain strong prior information about the magnitude of the number they encode. In sentence similarity and reasoning tasks, failure to handle numbers causes as much as 29% of contradictions [DMRM08]. In other data-intensive tasks where numbers are abundant, like neural machine translation, they are masked to hide the translation

models’ inefficiency in dealing with them [ML09]. There is a need for NLP models to better process numbers and understand their numeric properties. A technique called deterministic, independent-of-corpus (DICE) is proposed to mitigate the severe ineffectiveness of NLP models to deal with numbers. DICE embeddings [SSS+20] effectively capture properties of numeracy by taking the absolute difference between any two numbers, and incorporating them as part of a novel loss function. [SSW+22].

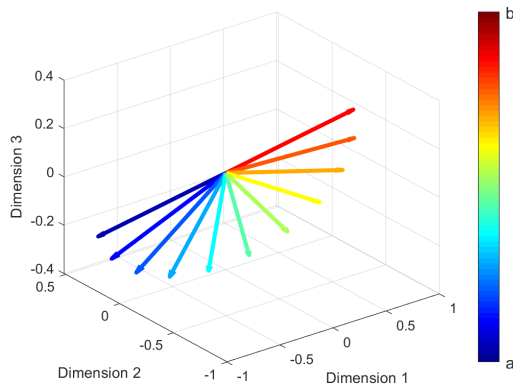


FIGURE 1.3: Proposed DICE embeddings.

$$d_n(x, y) = |x - y| \tag{1.3}$$

where  $d_n$  represents the absolute distance between any two numbers  $x$  and  $y$  captured by the proposed DICE method.

### 1.3 Feedback-based NLP

On feedback-based NLP, multi-task learning [Car97] has been shown to improve the performance of multiple related tasks through cross-task knowledge transfer using just one model, which also drastically improves parameter efficiency [HXTS16, KGS+17]. In the case where

the objective is improving specific target tasks, the use of auxiliaries has demonstrated substantial benefits, for example, in sequence-to-sequence learning [LLS<sup>+</sup>15], and question answering [MKXS18]. While there are numerous efforts that attempted to increase the complexity of multi-task models to match the performance through architectural addition, not a lot of emphasis has been placed on harnessing the potential of a model through the exploration of task sampling weights. A task-aware feedback-based model is developed by seeing a weighted combination of different tasks in a multitask setup as a sampling problem. [STL<sup>+</sup>21].

Detecting Out-of-Distribution (OOD) [GSS14, HG16, YZLL21] samples is vital for developing reliable machine learning systems for various industry-scale applications of natural language understanding (NLP) including intent understanding in conversational dialogues [ZCH20, LMS<sup>+</sup>17], language translation [DL11], and text classification [AZ12]. For instance, a language understanding model deployed to support a chat system for medical inquiries should reliably detect if the symptoms reported in a conversation constitute an OOD query so that the model may abstain from making an incorrect diagnosis. Motivated by the above limitations, a framework called POsthoc pseudo Ood REgularization (POORE) that generates pseudo-OOD data using the trained classifier and the In-Distribution (IND) samples is proposed. As opposed to methods that use outlier exposure, the framework doesn't rely on any external OOD set. Moreover, POORE can be easily applied to already deploy large-scale models trained on a classification task, without requiring to re-train the classifier from scratch. The model uses a Mahalanobis-based context masking scheme for generating pseudo-OOD samples that can be used during the fine-tuning. A new Pseudo Ood Regularization (POR) loss that maximizes the distance between IND and generated

pseudo-OOD samples to improve the OOD detection is also introduced.

Inspired by these possibilities mentioned above, the dissertation makes several contributions to the structure and feedback-based NLP. Chapter 2 discusses syntax-infused attention-based NLP models and their effects on translation. The syntax-infused BERT, in which only an encoder is involved, is also evaluated on a number of downstream tasks. Chapter 3 compares and contrasts the abilities of RNNs and Transformers to observe syntax and how word order affects translation across a variety of languages. Chapter 4 discusses the proposed method for incorporating numerical structure into the word embeddings and its effects on a number of numeracy tasks. Chapter 5 discusses the potential benefits of adopting numeracy as an entity and how it could be used in named entity recognition. Chapter 6 deals with feedback-based models, where a supporting reinforcement learning model weights different tasks to improve the performance of the target task, and Chapter 7 uses feedback signals to develop an efficient masking strategy to enhance the accuracy of OOD detection on NLP models.

## Syntax-infused Transformers and BERT

Attention-based deep learning models have demonstrated significant improvement over traditional algorithms in several NLP tasks. The Transformer, for instance, is an illustrative example that generates abstract representations of tokens that are input to an encoder based on their relationships to all tokens in a sequence. While recent studies have shown that such models are capable of learning syntactic features purely by seeing examples, we hypothesize that explicitly feeding this information to deep learning models can significantly enhance their performance in many cases. Leveraging syntactic information like part of speech (POS) may be particularly beneficial in limited-training-data settings for complex models such as the Transformer. In this work, we verify this hypothesis by infusing syntactic knowledge into the Transformer. We find that this *syntax-infused Transformer* achieves an improvement of 0.7 BLEU when trained on the full WMT '14 English to German translation dataset and a maximum improvement of 1.99 BLEU points when trained on a fraction of the dataset. In addition, we find that the incorporation of syntax into BERT fine-tuning outperforms BERT<sub>BASE</sub> on all downstream tasks from the GLUE

benchmark, including an improvement of 0.8% on CoLA.

## 2.1 Introduction

The Transformer relies on a significant amount of data and extensive training to accurately pick up on syntactic and semantic relationships [JSS<sup>+</sup>19]. Previous studies have shown that an NLP model’s performance improves with the ability to learn the underlying grammatical structure of a sentence [KDH<sup>+</sup>18,LDG16,CHCC17]. It has been shown that simultaneously training models for machine translation, part of speech (POS) tagging, and named entity recognition provide a slight improvement over baseline on each task for small datasets [NC17]. Inspired by these previous efforts, we propose to utilize external knowledge, namely the syntactic features inherent in natural language sequences, to enhance the performance of the Transformer model.

We suggest a modification to the embeddings fed into the Transformer architecture that allows the tokens input to the encoder to attend not only to other tokens but also syntactic features including POS, case (upper or lower), and subword position [SH16]. Like POS, case is a categorical feature that can allow the model to distinguish common words from important ones. Subword tags can help bring cohesion among subwords of a complex word (say ‘amal’, ‘ga’, ‘mation’ of “amalgamation”) so that their identity as a unit is not compromised by tokenization. These features are identified using a separate model (for POS) or are directly specified (for case and subword position) and are appended to the one-hot vector encoding for each token. For example, the feature-augmented subword tokens of ‘amalgamation’ would look like: ‘amal,NOUN,0,S’, ‘ga,NOUN,0,M’, ‘mation,NOUN,0,E’. Here, ‘NOUN’ indicates the POS; 0 indicates lowercase; and ‘S’, ‘M’, and ‘E’ indicates that the subwords occur at the start, middle, and end of the word, respectively. Embeddings

for the tokens and their features are learned jointly during the training. As the embeddings pass the Transformer layers, the representation for each token is synthesized using a combination of subword token and syntactic features.

We apply our approach to English to German (EN-DE) translation on the WMT '14 dataset and demonstrate that the BLEU score of the feature-rich *syntax-infused Transformer* uniformly outperforms the baseline Transformer as a function of training data size. We also evaluate the proposed model on two low-resource languages – Korean and Sinhala – where the source translation language is always English. Finally, we experiment with this modification of embeddings on the BERT<sub>BASE</sub> model on several General Language Understanding Evaluation (GLUE) benchmarks and observe considerable improvement in performance on all the tasks. Examining the attention weights learned by the proposed model indicates that the attention is more dispersed than the baseline model.

To summarize, our contributions are as follows:

1. We propose a modification to the trainable embeddings of the Transformer model, incorporating external syntax information, including POS, case, and subword position.
2. We demonstrate superior performance on the WMT '14 EN-DE machine translation task and for low-resource machine translation.
3. We infuse pre-trained BERT<sub>BASE</sub> embeddings with syntax information and find that our model outperforms BERT<sub>BASE</sub> on all of the GLUE benchmark tasks.

## 2.2 Related Work

Previous works have sought to improve the self-attention module to aid NLP models. For instance, [YTW<sup>+</sup>18] introduced a Gaussian bias to model locality, to enhance model ability to capture local context while also maintaining the long-range dependency. Instead of absolute positional embeddings, [SUV18] experimented with relative positional embeddings or distance between sequences and found that it led to a significant improvement in performance.

Adding linguistic structure to models like the Transformer can be thought of as a way of improving the attention mechanism. The POS and subword tags act as a form of relative positional embedding by enforcing the sentence structure. [LTY<sup>+</sup>18] encourages different attention heads to learn about different information like position and output representation by introducing a disagreement regularization. To model the local dependency between words more efficiently, [IC17] introduced distance between words and incorporated it into the self-attention.

Previous literature also has sought to incorporate syntax into deep learning NLP models. [BTA<sup>+</sup>17] used syntax dependency tree on a bidirectional RNN on translation systems by modeling the trees using Graph Convolutional Networks (GCNs) [KW16]. Incorporating syntax information by linearizing parse trees and adding a syntax-based distance constraint on the attention module helped significantly in Chinese-English and English-German translation [LXT<sup>+</sup>17].

Finally, [CH19] has incorporated source-side syntax on Transformer encoders in a multi-task setup with parse trees. However, in this approach, the Transformer struggles to learn to parse and decode the syntactic sequences and only showed performance gains in low-

resource languages. We append the syntactic information as sequences and decode only the target sequence, thus not complicating the Transformer’s job. These works affirm that adding syntax information can help the NLP models translate better from one language to another and achieve better performance measures.

## 2.3 Background

### 2.3.1 Baseline Transformer

The Transformer consists of encoder and decoder modules, each containing several subunits that act sequentially to generate abstract representations for words in the source and target sequences [VSP<sup>+</sup>17]. For all  $m \in \{1, 2, \dots, M\}$ , where  $M$  is the length of the source sequence, the encoder embedding layer first converts tokens  $\mathbf{x}_m$  into embeddings  $\mathbf{e}_m$ :  $\mathbf{e}_m = \mathbf{E}\mathbf{x}_m$  where  $\mathbf{E} \in \mathbb{R}^{D \times N}$  is a trainable matrix with column  $m$  constituting the embedding for token  $m$ ,  $N$  is the total number of tokens in the shared vocabulary, and  $\mathbf{x}_m \in \{0, 1\}^N : \sum_i x_{mi} = 1$  is a one-hot vector corresponding to token  $m$ . These embeddings are passed sequentially through six encoder subunits. Each of these subunits features a self-attention mechanism, that allows tokens in the input sequence to be represented as a combination of all tokens in the sequence. Attention is accomplished using three sets of weights: the key, query, and value matrices ( $\mathbf{K}$ ,  $\mathbf{Q}$ , and  $\mathbf{V}$ , respectively). The key and query matrices interact to score each token in relation to other tokens, and the value matrix gives the weights to which the score is applied to generate output embedding of a given

token. Stated mathematically,

$$\begin{aligned}
 \mathbf{K} &= \mathbf{H}\mathbf{W}_K \\
 \mathbf{Q} &= \mathbf{H}\mathbf{W}_Q \\
 \mathbf{V} &= \mathbf{H}\mathbf{W}_V \\
 \mathbf{A} &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{\rho}}\right)\mathbf{V}
 \end{aligned}
 \tag{2.1}$$

where  $\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \dots \ \mathbf{h}_M]^\top \in \mathbb{R}^{M \times D}$  are the  $D$ -dimensional embeddings for a sequence of  $M$  tokens indexed by  $m$ ;  $\mathbf{W}_K$ ,  $\mathbf{W}_Q$ , and  $\mathbf{W}_V$  all  $\in \mathbb{R}^{D \times P}$  are the projection matrices for keys, queries, and values, respectively;  $\rho$  is a scaling constant (here, taken to be  $P$ ) and  $\mathbf{A} \in \mathbb{R}^{M \times P}$  is the attention-weighted representation of each token. Note that these are subunit-specific – a separate attention-weighted representation is generated by each subunit and passed on to the next. Moreover, for the first layer,  $\mathbf{h}_m := \mathbf{e}_m$ .

The final subunit then passes its information to the decoder, which also consists of six identical subunits that behave similarly to those of the encoder. One key difference between the encoder and decoder is that the decoder not only features self-attention but also cross-attention; thus, when generating new words, the decoder pays attention to the entire input sequence as well as to previously decoded words.

### 2.3.2 BERT

The token embeddings learned by the Transformer encoder can also be fine-tuned to perform a number of different downstream tasks. Bidirectional encoder representations of Transformers (BERT) [DCLT18] is an extension of the Transformer model that allows for such fine-tuning. The BERT model is essentially a Transformer encoder (with number of layers  $l$ , embedding dimension  $D$ , and number of attention heads  $\alpha$ ) which is pre-trained us-

ing two methods: masked language modeling (MLM) and next-sentence prediction (NSP). Subsequently, a softmax layer is added, allowing the model to perform various tasks such as classification, sequence labeling, question answering, and language inference.

## 2.4 Knowledge-Infused Models

### 2.4.1 *Syntax-infused Transformer*

To aid the Transformer in acquiring and utilizing syntactic information for better translation, we (i) employ a pre-trained model<sup>1</sup> with high accuracy to tag words in the source sequence with their POS, (ii) identify the case of each word, and (iii) identify the position of each subword relative to other subwords that are part of the same word (subword tagging). We then append trainable syntax embedding vectors to the token embeddings, resulting in a combined representation of syntactic and semantic elements. Specifically, each word in the source sequence is first associated with its POS label according to syntactic structure. We then assign each subword token the POS label of the word from which it originated. For example, if the word **sunshine** is broken up into subwords **sun**, **sh**, and **ine**, each subword token would be assigned the POS NOUN. The POS embeddings  $\mathbf{f}_m^P \in \mathbb{R}^d$  for each token (indexed by  $m$ ) are then extracted from a trainable embedding matrix using a look-up table.

In a similar manner, we extract case and subword position features. For case, we use a binary element  $z_m^c \in \{0, 1\}$  to look up a feature embedding  $\mathbf{f}_m^c \in \mathbb{R}^d$  for each subword, depending on whether the original word is capitalized. For subword position, we use a categorical element  $z_m^s \in \{S, M, E, O\}$  to identify a feature embedding  $\mathbf{f}_m^s \in \mathbb{R}^d$  for each subword depending on whether the subword is at the start ( $S$ ), middle ( $M$ ), or end ( $E$ )

---

<sup>1</sup> <https://spacy.io/>

of the word; if the subword comprises the full word, it is given a tag of  $O$ . These are then added onto the POS embedding and concatenated with the token embeddings  $\mathbf{e}_m \in \mathbb{R}^{D-d}$  to create a combined embedding (see Figure 2.1a). Mathematically, in the input stage,  $\mathbf{h}_m$  becomes:

$$[\mathbf{e}_m^\top \mathbf{f}_m^\top]^\top = \mathbf{h}'_m \in \mathbb{R}^D$$

where  $\mathbf{f}_m = \mathbf{f}_m^P + \mathbf{f}_m^c + \mathbf{f}_m^s \in \mathbb{R}^d$  is the learned embedding for the syntactic features of subword  $m$  in the sequence of  $M$  subwords. The augmented embeddings are then passed through the Transformer’s multi-headed attention, layer normalization, and feedforward modules, resulting in a syntax-infused hidden state of each encoder layer:

$$\begin{aligned} \tilde{\mathbf{A}} &= [\mathbf{A}_0 \cdots \mathbf{A}_7] \mathbf{W}^O \\ \tilde{\mathbf{H}} &= [\mathbf{H}' \cdots \mathbf{H}'] \\ \tilde{\mathbf{Z}} &= \text{LayerNorm}(\tilde{\mathbf{A}} + \tilde{\mathbf{H}}) \\ \mathbf{Z}_l &= \text{LayerNorm}(\tilde{\mathbf{Z}} + \text{FeedForward}(\tilde{\mathbf{Z}})) \end{aligned} \tag{2.2}$$

where  $\tilde{\mathbf{H}}$  are the syntax-augmented embeddings, repeated eight times (once for each attention head);  $\tilde{\mathbf{A}}$  is the concatenation of the attention-weighted representation of the output of the eight heads; and  $\mathbf{Z}_l$  is the output of layer  $l$ . LayerNorm and FeedForward denote the layer normalization function and feedforward neural network layers, respectively, and are applied separately for each head. This is repeated  $L$  times in  $L$  layers, with the output of the previous layer treated as the input to the next. The final syntax-infused encoder output  $\mathbf{Z}_L$  is attended to by the decoder, aiding the autoregressive text generation process

in the final layer of the Transformer:

$$\begin{aligned} \mathbf{z}'_{L,t} &= f(\mathbf{Z}_L, w_{1:t-1}) \\ \text{Scores}(t) &= \text{LayerNorm}(\mathbf{z}'_{L,t})\mathbf{W}_V \\ P(w_t|w_{1:t-1}, \mathbf{z}'_{L,t}) &= \text{softmax}(\text{Scores}(t)) \end{aligned} \tag{2.3}$$

where  $f$  is the decoder, which undergoes the same process outlined in equation 2.2, and comprises cross-attention with the encoder output and a causal self-attention (parameters of the decoder omitted for conciseness).  $t = 1, \dots, T$  indexes decoded tokens  $w_t$ , and  $\mathbf{W}_V$  is a linear projection matrix mapping the decoder output to the vocabulary size. The scores for output tokens are now a function of  $\mathbf{z}'_{L,t}$ , which is aware of the source syntax.

#### 2.4.2 Syntax-infused BERT

Adding syntactic features to the BERT model is a natural extension of the above modification to the Transformer. For a given token, its input representation is constructed by summing the corresponding BERT token embeddings with POS embeddings (see Figure 2.1b). Mathematically, the input tokens  $\mathbf{h}'_m \in \mathbb{R}^D$  are given by  $\mathbf{h}'_m = \mathbf{e}_m + \mathbf{f}_m^P$ , where  $\mathbf{e}_m$  is the BERT token embedding and  $\mathbf{f}_m^P$  is the POS embedding for token  $m$ . For single sequence tasks,  $m = 1, 2, \dots, M$ , where  $M$  is the number of tokens in the sequence; while for paired sequence tasks,  $m = 1, 2, \dots, M_1 + M_2$ , where  $M_1$  and  $M_2$  are the number of tokens in each sequence. As is standard with BERT, for downstream classification tasks, the final embedded representation  $\hat{\mathbf{y}}_{CLS}$  of the first token (denoted as [CLS]) is passed through a softmax classifier to generate a label. We modify the BERT<sub>BASE</sub> model using this approach and denote it as BERT<sub>BASE + POS</sub>.

Table 2.1: BLEU scores for different proportions of the data for baseline Transformer vs syntax-infused Transformer for the EN-DE task on newstest2014.

Data Fraction	Number of Sentences	Baseline Transformer	Syntax-infused Transformer
1%	45k	1.10	<b>1.67</b>
5%	225k	8.51	<b>10.50</b>
10%	450k	16.28	<b>17.28</b>
25%	1.1M	22.72	<b>23.24</b>
50%	2.25M	25.41	<b>25.74</b>
100%	4.5M	28.94	<b>29.64</b>

## 2.5 Datasets and Experimental Details

We train our syntax-infused model on the WMT '14 EN-DE (German) dataset, which consists of 4.5M training sentence pairs. Validation is performed on newstest2013 (3000 sentence pairs), and testing is on the newstest2014 dataset (2737 sentence pairs, [ZTS19]). For low-resource machine translation (1M sentence pairs or less), we employ the OpenSubtitles EN-KO (Korean) and EN-SI (Sinhala) datasets [LT16]. The latter was found by [GCO<sup>+</sup>19] to be a low-resource language with morphology and syntax significantly different from that of English. We subsample these datasets from 1.3M and 601K sentence pairs, respectively, to 500K sentence pairs, and train on 90%, validate on 9%, and test on 1%. One of the key reasons to incorporate POS features to EN sequences is that parsers that infer syntax from EN sentences are typically trained on a greater number and variety of sentences and are therefore more robust than parsers for other languages. Unlike [CH19], in our experiments, we always translate sequences from EN to other languages as the objective is to improve the translation performance and not to decode the POS sequences.

Table 2.2: BLEU scores improvements on low resource languages including Sinhala and Korean for the baseline and syntax-infused models.

	Transformer		RNN	
Lang.	Baseline	Syntax	Baseline	Syntax
EN-SI	11.12	<b>11.73</b>	8.24	7.95
EN-KO	3.87	<b>4.19</b>	2.86	1.80

### 2.5.1 Machine translation

For EN-DE, we train both the baseline and syntax-infused Transformer for 100,000 steps. All hyperparameter settings of the baseline Transformer, including embedding dimensions of the encoder and decoder, match those of [VSP<sup>+</sup>17]. We train the syntax-infused Transformer model using 512-dimensional embedding vectors. In the encoder,  $D = 492$  dimensions are allocated for subword token embeddings while  $d = 20$  for feature embeddings (chosen by hyperparameter tuning). In the decoder, all 512 dimensions are used for subword token embeddings (since we are decoding words, not word-POS pairs).

The model architecture consists of six encoder and six decoder layers, with eight heads for multi-headed attention. Parameters are initialized from a Glorot uniform distribution [GB10]. We use a dropout rate of 0.1 and batch size of 4096. We utilize the Adam optimizer to train the model with  $\beta_1 = 0.9$  and  $\beta_2 = 0.998$ ; gradients are accumulated for two batches before updating parameters. A label-smoothing factor of 0.1 is employed.

The training settings are exactly the same for low-resource translation except that we train the Transformer models for 50,000 steps owing to the reduced data complexity. The dimension  $d$  of features  $\mathbf{f}_m$  is chosen to be 20 by doing a grid search over the range of 8 to 64.

### 2.5.2 *Natural language understanding*

The General Language Understanding Evaluation (GLUE) benchmark [WSM<sup>+</sup>18] is a collection of different natural language understanding tasks evaluated on eight datasets: Multi-Genre Natural Language Inference (MNLI), Quora Question Pairs (QQP), Question Natural Language Inference (QNLI), Stanford Sentiment Treebank (SST-2), the Corpus of Linguistic Acceptability (CoLA), the Semantic Textual Similarity Benchmark (STS-B), Microsoft Research Paraphrase Corpus (MRPC), and Recognizing Textual Entailment (RTE). For a summary of these datasets, see [DCLT18]. We use POS as the syntactic feature for BERT for these tasks. Aside from the learning rate, we use identical hyperparameter settings to fine-tune both the BERT<sub>BASE</sub> and BERT<sub>BASE + POS</sub> models for each task. This includes a batch size of 32 and 3 epochs of training for all tasks. For each model, we also choose a task-specific learning rate among the values  $\{5, 4, 3, 2\} \times 10^{-5}$ , which is standard for BERT<sub>BASE</sub>.

## 2.6 Experimental Results

### 2.6.1 *Machine translation*

We evaluate the impact of infusing syntax into the baseline Transformer on the WMT '14 EN-DE translation task as well as on the low resource EN-SI and EN-KO language pairs. There are multiple ways to incorporate feature embeddings into the subword token embeddings, such as direct summation, vector projection, or concatenation. For a fair comparison to the baseline Transformer, we fix a total of 512 dimensions for representing both the token embeddings and feature embeddings. One important tradeoff is that as the dimensionality of the syntax information increases, the dimensionality for token embed-

Table 2.3: Translation examples of baseline Transformer vs. syntax-infused Transformer on the EN-DE dataset. The text highlighted in blue represents words correctly predicted by the syntax-infused model but not by the baseline Transformer.

Reference	Baseline Transformer	Syntax-infused Transformer
Parken in Frankfurt könnte bald empfindlich teurer werden .	Das Personal war sehr freundlich und hilfsbereit .	Parken in Frankfurt könnte bald spürbar teurer sein .
Die zurückgerufenen Modelle wurden zwischen dem 1. August und 10. September hergestellt .	Zwischen August 1 und September 10.	Die zurückgerufenen Modelle wurden zwischen dem 1. August und 10. September gebaut
Stattdessen verbrachte Bwelle Jahre damit , seinen Vater in überfüllte Kliniken und Hospitäler zu begleiten , um dort die Behandlung zu bekommen , die sie zu bieten hatten .	Stattdessen verbrachte Bwelle Jahre damit , seinen Vater mit überfüllten Kliniken und Krankenhäusern zu beherbergen .	Stattdessen verbrachte Bwelle Jahre damit , seinen Vater zu überfüllten Kliniken und Krankenhäusern zu begleiten , um jede Behandlung zu bekommen , die sie bekommen konnten .
Patek kann gegen sein Urteil noch Berufung ein legen .	Patek kann noch seinen Satz an rufen .	Patek mag sein Urteil noch Berufung ein legen .

dings decreases. Since POS, case, and subword tags have only a limited number of values they can take, dedicating a high dimensionality to each feature proves detrimental. We find that the total feature dimension for which the gain in BLEU score is maximized is 20 (found through grid search). This means that (1) each feature embedding dimension can be allocated to 20 and summed together or (2) the feature embeddings can be concatenated to each other such that their total dimension is 20. Therefore, in order to efficiently learn the feature embeddings while also not sacrificing the token embedding dimensionality, we find that summing the embeddings for all three different features of  $d = 20$  and concatenating the sum to the subword token embeddings of  $D = 492$  gives the maximum performance on translation.

We report results on the WMT '14 EN-DE translation task in Table 2.1. We vary the

Table 2.4: GLUE test results scored using the GLUE evaluation server. The number below each task denotes the number of training examples. The scores in bold denote the tasks for which BERT<sub>BASE + POS</sub> outperforms BERT<sub>BASE</sub>.

System	MNLI 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>BASE + POS</sub>	84.4/ <b>83.6</b>	<b>71.4</b>	<b>90.8</b>	<b>93.9</b>	<b>52.9</b>	<b>86.0</b>	<b>89.0</b>	<b>66.9</b>	<b>79.9</b>

Table 2.5: Examples of randomly chosen sentences from the RTE dataset (for evaluation of entailment between pairs of sentences) that were classified by BERT<sub>BASE + POS</sub> but not by BERT<sub>BASE</sub>.

Sentence 1	Sentence 2	True label
The Qin (from which the name China is derived) established the approximate boundaries and basic administrative system that all subsequent dynasties were to follow .	Qin Shi Huang was the first Chinese Emperor .	Not entailment
Steve Jobs was attacked by Sculley and other Apple executives for not delivering enough hot new products and resigned from the company a few weeks later.	Steve Jobs worked for Apple.	Entailment

proportion of data used for training and observe the performance of both the baseline and syntax-infused Transformer. The syntax-infused model markedly outperforms the baseline model, offering an improvement of 0.57, 1.99, 1, 0.52, 0.33, and 0.7 points, respectively, when trained on 1, 5, 10, 25, 50, and 100% of the full training data. It is notable that the syntax-infused model translates the best relative to the baseline when only a fraction of the dataset is used for training. Specifically, the maximum improvement is 1.99 BLEU points when only 10% of the training data is used. This shows that explicit syntax information is most helpful under limited training data conditions. As shown in Figure 2.2(a)-(b), the syntax-infused model is better able to capture connections between tokens that are far apart

yet semantically related, resulting in improved translation performance. In addition, Table 2.3 shows a set of sample German predictions made by the baseline and syntax-infused Transformer.

In Table 2.2, we compare the performance of baseline and syntax-infused Transformers on low resource languages, namely Korean and Sinhala. These languages are often underrepresented in the NLP literature and fall under categories of languages with limited training data. Results confirm the effectiveness of our methodology of adding syntax to source sequences to improve efficiency, irrespective of the training data size. In addition, while we find there are improvements in BLEU scores with the syntax-infused Transformer, Table 2.2 also shows the effect of the addition of syntax to RNN-based networks for the same translation tasks leads to a drop in performance. For this experiment, we employed a 2-layer LSTM encoder and 2-layer LSTM decoder with 500 hidden states each and augmented encoder tokens with syntax features in the same way as for the Transformer. [LDG16] confirms our observation that recurrent structures struggle to learn the subject-verb agreement and that better architectures may be required to retain the complex syntactic structures present in the languages. On the other hand, Transformers have been known to learn syntax with sufficient training and an increased amount of data. Studies [WWL19b] have also shown that explicit syntax supervision in Transformers yields superior results.

### *2.6.2 Natural language understanding*

Results obtained for the  $BERT_{BASE+POS}$  model on the GLUE benchmark test set are presented in Table 2.4.  $BERT_{BASE+POS}$  outperforms  $BERT_{BASE}$  on all tasks, with the exception of matched MNLI. The improvements range from marginal to significant, with a maximum improvement of 0.8 points of the POS model over  $BERT_{BASE}$  on CoLA. Fittingly,

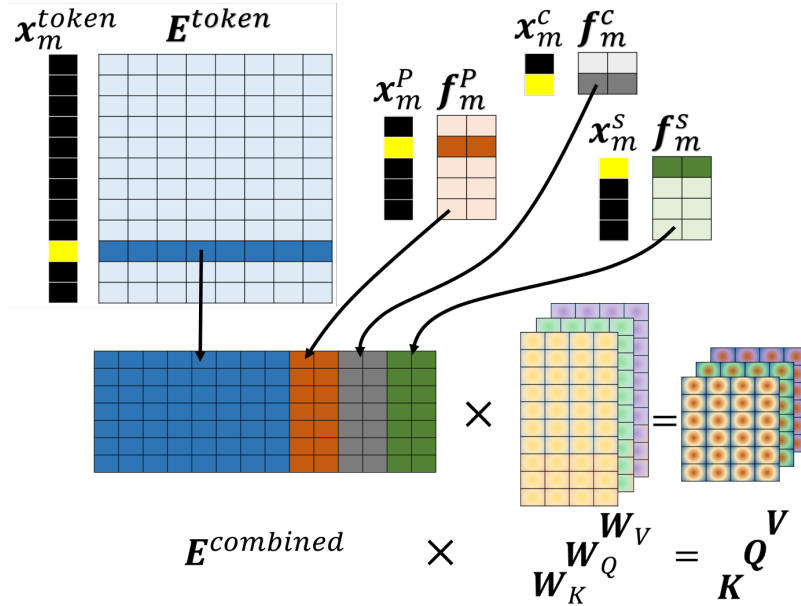
CoLA is a task which assesses the linguistic structure of a sentence, which is explicitly informed by POS embeddings. Moreover,  $\text{BERT}_{\text{BASE} + \text{POS}}$  outperforms  $\text{BERT}_{\text{BASE}}$  on tasks that are concerned with evaluating semantic relatedness. For examples of predictions where  $\text{BERT}_{\text{BASE} + \text{POS}}$  performs better than  $\text{BERT}_{\text{BASE}}$  made on the RTE dataset, see Table 2.5.

## 2.7 Conclusions

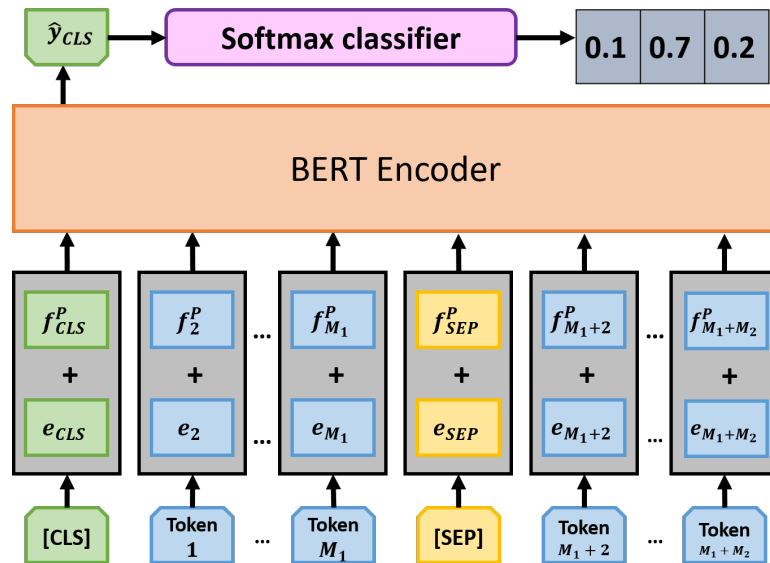
Infusing external syntax into NLP models has proven to be an effective method to improve performance on generation as well as classification tasks. In this work, we have infused the Transformer and BERT models with syntactic features including POS, word case, and subword position using a simple concatenation of token embeddings and trainable feature embeddings from external knowledge. We conducted experiments on the large-scale WMT '14 EN-DE translation task and low-resource EN-KO and EN-SI OpenSubtitles datasets. We found significant improvements in EN-DE translation, especially as the size of the training dataset is reduced. These results were confirmed in the low-resource setting, suggesting that our method performs well regardless of the training data size, a feature that makes it less reliant on the training data. We then modified BERT by adding trainable syntactic embeddings directly to the input and found that our  $\text{BERT}_{\text{BASE} + \text{POS}}$  model performs better than baseline on a number of GLUE downstream tasks. Thus, we have proven both text generation and classification applications benefit from knowledge injection.

While many existing works either suggest that Transformers are capable of learning syntax implicitly or attempt to train Transformers to predict syntax in a multi-task setting, our results suggest that, when available, it is preferable to directly incorporate syntactic features as inputs to the NLP model to improve performance. As a next step, we plan

to study the alignment of tokens between source and target sequences to determine the effect of adding syntax into the model. In addition, we plan to investigate syntax infusion into other NLP tasks such as co-reference resolution. In this context, syntactic information could be helpful in matching tokens to their antecedents by virtue of parts of speech, for instance.



(a) Syntax-infused Transformer



(b) Syntax-infused BERT

FIGURE 2.1: (a) Formation of attention matrices ( $\mathbf{K}$ ,  $\mathbf{Q}$ , and  $\mathbf{V}$ ) with syntactic information. The left column shows the token embedding matrix; the embedding matrices for the various features are shown on top. Embeddings for the features are either concatenated or summed (denoted by  $\oplus$ ) and finally, concatenated to the token embeddings. Multiplication with learned weights results in  $\mathbf{K}$ ,  $\mathbf{Q}$ , and  $\mathbf{V}$ . The attention matrices are double shaded to indicate the mix of token and syntax information. (b) The  $BERT_{BASE+POS}$  model. Token embeddings are combined with trainable POS embeddings and fed into the BERT encoder. The final embedding of the [CLS] token is fed into a softmax for downstream classification tasks.

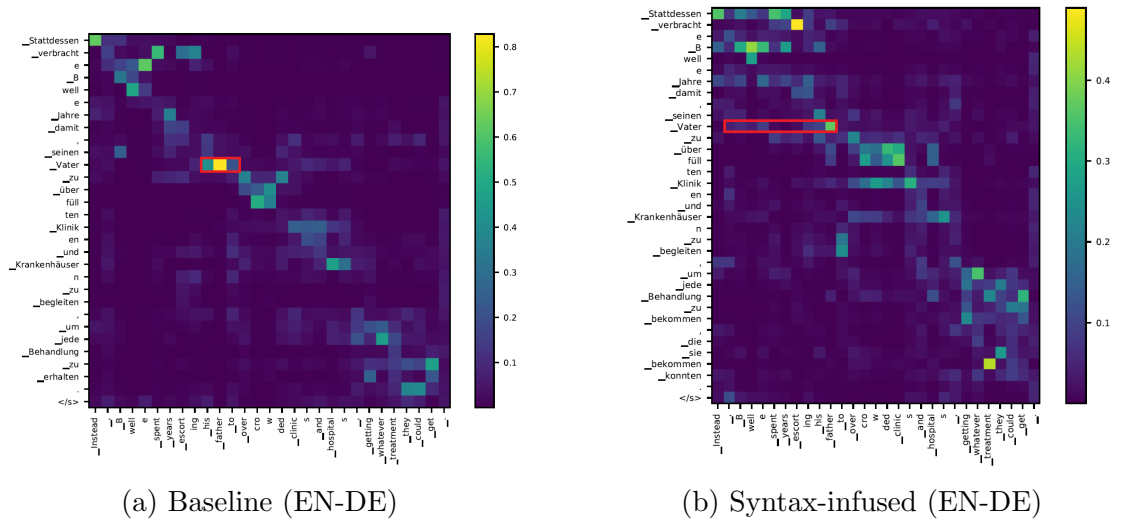


FIGURE 2.2: Comparison of attention for example sentences translated by baseline and POS Transformer models (obtained from the last layer). Rows depict the attention score for a given target subword to each of the subwords in the source sequence. In syntax-infused models for EN-DE translation, we find that attention is more widely distributed across subwords. For instance, the subword “Vater” (the German word for “father”) attends mostly to the nearby subwords “his” and “father” in the base model while “Vater” also attends to the more distant words “Bwelle” (a person) and “escorting” in the syntax-infused model. This suggests that the syntax-infused model is able to better connect disparate parts of a sentence to aid translation. Note that the number of rows in the baseline and syntax-infused Transformer are different because each produces different predictions.

## Syntax and Lexical Semantics

Neural machine translation (NMT) systems aim to map text from one language into another. While there are a wide variety of applications of NMT, one of the most important is the translation of *natural language*. A distinguishing factor of natural language is that words are typically ordered according to the rules of the grammar of a given language. Although many advances have been made in developing NMT systems for translating natural language, little research has been done on understanding how the word ordering of and lexical similarity between the source and target language affect translation performance. Here, we investigate these relationships on a variety of low-resource language pairs from the OpenSubtitles2016 database, where the source language is English, and find that the more similar the target language is to English, the greater the translation performance. In addition, we study the impact of providing NMT models with part of speech of words (POS) in the English sequence and find that, for transformer-based models, the more dissimilar the target language is from English, the greater the benefit provided by POS.

### 3.1 Introduction

Neural machine translation (NMT) systems map text from one language into another via a neural network. Several approaches to NMT have been developed, commonly consisting of an encoder-decoder architecture and an attention mechanism [SVL14,BCB14,LPM15]. The encoder seeks to extract a representation for the source sequence that captures all relevant semantics in the sequence. The decoder then utilizes this representation to generate a sequence of words, which is the translation. Attention allows the decoder to weight individual tokens in the source sequence depending on their importance to the word being generated. The Transformer [VSP<sup>+</sup>17], a more complex architecture that employs multi-headed self- and cross-attention<sup>1</sup>, leading to a new state-of-the-art, has led to an unprecedented wave of research in NMT.

While these systems have shown a great deal of promise, relatively little has been done to understand in detail how the *lexical semantics* of natural language, including *lexical similarity* and *word order* (e.g., subject-verb-object, or SVO), of the source and target language affects translation performance. For instance, [JSL<sup>+</sup>17a] apply Google NMT, an LSTM-based architecture, to multilingual translation, achieving zero-shot learning between related languages Portuguese  $\leftrightarrow$  Spanish and Korean  $\leftrightarrow$  Japanese. However, little discussion is provided as to how the syntactic features of these languages contribute to performance. Recently, [AJF19] utilized the Transformer architecture for multilingual translation on an in-house dataset consisting of 58 low-resource language pairs. While they acknowledge that the diversity of linguistic features can induce a bottleneck due to limited model capacity, they do not make any effort to analyze similarities between the source and

---

<sup>1</sup> Among other features, including positional encoding and layer normalization

target languages or to prune the languages on which their multilingual models are trained.

Hence, in this paper, we provide two main contributions. **First:** we perform an empirical study of translation from English into 15 languages of various word orders and degrees of lexical similarity to English. We utilize both LSTM- and Transformer-based models for our study. **Second:** we compare and contrast how explicit supervision with part of speech (POS), a grammatical feature closely tied to word order, of source-side tokens affects the translation performance of these respective models. With insight into these trends, we hope to further research on multilingual translation by highlighting the importance of accounting directly for differences in lexical semantics during model development.

## 3.2 Methods

### 3.2.1 Models

For our experiments, we utilize LSTM- and Transformer-based architectures and refer the reader to [SVL14] and [VSP<sup>+</sup>17] for details. For each parallel corpus, words are first tokenized into subwords with byte pair encoding [SHB15] using a common vocabulary for source and target languages. The LSTM and Transformer baselines allow us to study general trends in performance as a function of word order and lexical similarity. We then introduce two related models denoted LSTM<sub>POS</sub> and Transformer<sub>POS</sub> to which we directly provide part of speech (POS) of each word in the source sequence, obtained using `spaCy`<sup>2</sup>. As POS is closely tied to word order, we hypothesize that providing POS of the source sequence to the decoder will improve alignment, attention, and ultimately, text generation, especially for languages whose word order is substantially different from the source. We assign the POS of each word to its subword tokens and append trainable POS embedding

---

<sup>2</sup> <https://spacy.io/>

Table 3.1: Target languages selected from OpenSubtitles2016 database, sorted first by word order then inversely by normalized Levenshtein distance to English, whose word ordering is SVO. Some languages have more than one dominant word order and are listed as such, with the most dominant appearing first. For each language, the number of training and testing examples is listed, along with the BLEU score achieved by the baseline LSTM and Transformer models.

Target Language	Word Order	Levenshtein distance	Num. train	Num. test	LSTM: BLEU score	Transformer: BLEU score
Sinhala (SI)	SOV	0.642	540990	6075	11.36	12.76
Bengali (BN)	SOV	0.632	372240	4138	11.60	13.41
Hindi (HI)	SOV	0.632	83700	946	22.26	23.80
Malayalam (ML)	SOV / Flexible	0.708	348120	3936	7.35	8.12
Korean (KO)	SOV / Flexible	0.468	1251990	14001	5.66	6.85
Basque (EU)	Flexible / SOV	0.407	725130	8137	14.98	16.60
Georgian (KA)	Flexible	0.621	177910	2077	10.57	11.79
Chinese (ZH_CN)	Flexible	0.519	450000	5000	6.84	7.73
Esperanto (EO)	Flexible / SVO	0.398	57960	729	11.31	12.65
Latvian (LV)	SVO / Flexible	0.416	467550	5248	17.63	19.71
Galician (GL)	SVO / Flexible	0.390	183150	2085	15.84	17.86
Ukrainian (UK)	SVO	0.539	789930	8857	11.46	13.61
French (FR)	SVO	0.386	450000	5000	22.53	23.95
German (DE)	SVO	0.383	450000	5000	19.43	20.57
Catalan (CA)	SVO	0.382	434250	4923	27.30	29.23

vectors to the subword token embeddings in a manner similar to [SH16,SSW<sup>+</sup>19].

### 3.2.2 Dataset

We translate from English into 15 target languages from the OpenSubtitles2016 database, which consists of human-annotated captions for movies and films [LT16]. Unlike other publicly available parallel corpora such as Europarl [Koe05], JRC-Acquis [SPW<sup>+</sup>06], or WIT<sup>3</sup> [CGF12], the OpenSubtitles2016 database contains data from languages with a more diverse range of word orderings and lexical similarity, allowing us to robustly test our hypothesis. Of the 15 target languages chosen (see Table 3.1), 12 (Sinhala, Bengali, Hindi, Malayalam, Korean, Basque, Georgian, Esperanto, Latvian, Galician, Ukrainian, and Catalan) are considered low-resource since they have roughly 1 million sentence pairs or less,

and 3 (Simplified Chinese, French, and German) are considered high-resource. We choose to focus primarily on low-resource languages as the performance of NMT systems has already achieved near human-level performance in data-rich settings [WSC<sup>+</sup>16, HAC<sup>+</sup>18]. Thus, low-resource language pairs stand to benefit the most from novel improvements to NMT architectures that harness specific features of the source and target languages. The remaining 3 languages were subsampled to 500K data points for consistency.

### 3.2.3 *Word order and lexical similarity*

We compare translation performance between our baselines and POS-augmented models as a function of two key features: *word order* and *lexical similarity*. For each target language, word order was obtained using Glottolog [HFH17], a professionally curated online catalog of the world’s languages. These are listed in Table 3.1. Target languages fall into three major categories: subject-object-verb (SOV), flexible, or subject-verb-object (SVO). For languages that are flexible in structure but which still possess a prevalent form, both “flexible” and the prevalent form are listed, with whichever is more dominant appearing first. This allows us to view the languages on a spectrum, with SOV languages on one end and SVO languages on the other – closest to English, whose word order is SVO.

We compute the lexical similarity between the source and target documents via normalized Levenshtein distance [JZ07]: the Levenshtein distance first computes the minimum number of character modifications (insertion, replacement, or deletions) that must be performed to map from the source to the target, and this value is then normalized by the average of the number of characters in the source and target. Levenshtein distance can be computed exactly in  $\mathcal{O}(nm)$  time, where  $n$  is the number of characters in the source and  $m$  is the number of characters in the target. This metric makes the implicit assumption that

two languages are lexically similar when (1) their vocabularies consist of similar phonemes and (2) the phonemes are composed of similar numbers of characters. This simplified definition affords us the ability to compare languages whose alphabets may have no common characters without having to resort to the manual transliteration of symbols into a common space of phonemes.

#### 3.2.4 Training details

We utilize the OpenNMT-py LSTM and Transformer implementations [KKD<sup>+</sup>17a] and specify identical training conditions for each model type when training models on all 15 corpora. Specifically, for the LSTM models, we train using a Titan XP GPU for 75,000 steps using an embedding dimension of 512, hidden state dimension of 512, batch size of 64, and a dropout rate of 0.1. For the Transformer models, we train for 75,000 steps using the same embedding and hidden state dimensions, batch size of 4096, and same dropout rate. For both models, POS embeddings were merged with subword token embeddings using a feature vector exponent of 0.7, resulting in roughly 5 to 8 dimensions being allocated for POS, depending on the corpus. We train all models with an Adam optimizer with  $\eta = 1$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-7}$ .

### 3.3 Results and Discussion

We evaluate translation performance with BLEU scores [PRWZ02] and report baseline results in Table 3.1 for each of the fifteen target languages. As expected, we find that the translation performance of both models drops as the word order of the target language differs more from that of English, which is SVO. We also find that translation performance decays roughly linearly with Levenshtein distance (Pearson’s  $r = -0.47$ ,  $p = 0.08$  for both

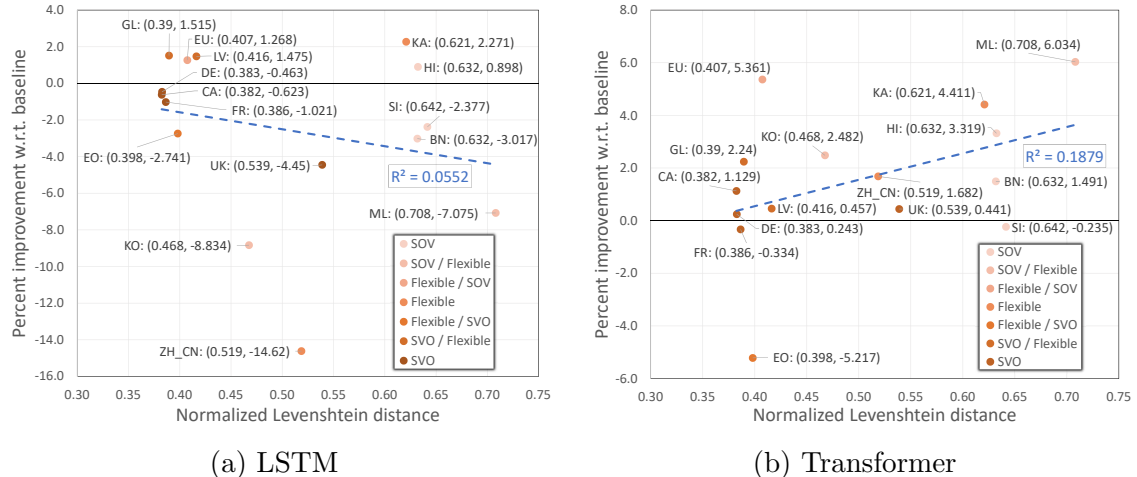


FIGURE 3.1: Performance improvements with POS models. Baseline BLEU scores given in Table 3.1. Shades of red indicate the word order of the target language. The dashed blue line-of-best-fit indicates the overall correlation.

models).

HI and UK are outliers within their respective word order groups. The reduced performance on EN→UK can be explained by the stark difference in the lexical similarity of UK to EN compared to FR, DE, and CA. For EN→HI, regressions of BLEU score against (1) number of testing samples and (2) number of unique English words in each of the 15 parallel corpora found no significant trends, ruling out biases (1) in the size of the test set or (2) due to overall vocabulary size. Thus, we believe the difference originates from unique aspects of HI that set it apart from SI and BN. For instance, conjugation of verbs in BN is much more subtle, often requiring changes of just a single syllable; in contrast, HI conjugations often require the addition of an extra word. In addition, in HI, both determiners and verbs are gendered (*e.g.*, “he eats the apple” is different from “she eats the orange”). The greater number of distinguishing factors reduces the overall entropy of predicting words in HI, leading to an increase in the BLEU score.

Figure 1 depicts the change in performance of each of these architecture styles when

POS is included as an input feature. Notably, the LSTM<sub>POS</sub> models generally perform worse compared to their baseline counterparts (mean difference of -2.51%;  $t$ -test,  $p = 0.06$ ) while the Transformer<sub>POS</sub> models perform significantly better (mean difference of +1.64%;  $t$ -test,  $p = 0.02$ ). Furthermore, the degradation of the LSTM<sub>POS</sub> performance worsens slightly as the word order of the target language becomes more different (MWW test,  $p = 0.15$ ) and as lexical similarity decreases (Pearson’s  $r = -0.24$ ,  $p = 0.40$ ). On the other hand, gains were seen by Transformer<sub>POS</sub> improve more and more as the disparity between source and target languages increases (MWW test,  $p = 0.02$  for word order; Pearson’s  $r = +0.43$ ,  $p = 0.11$  for lexical similarity). Thus, while the LSTM may be able to infer semantic relationships between distant tokens that are preserved during decoding, its reduced model capacity relative to the Transformer renders it inept at generating syntactically correct sequences as the ordering of target words begins to change. In contrast, the Transformer utilizes source POS features as anchor points to effectively learn the word ordering of source sequence (self-attention) and to perform better alignment during decoding (cross-attention).

### 3.4 Conclusion

In conclusion, we have demonstrated that both LSTMs and Transformers perform best at NMT of natural language when the source and target languages possess similar lexical semantics. In addition, incorporating POS as an input feature to the Transformer helps the model align words, especially when the source and target word orderings are severely mismatched. Future work will be towards incorporating these findings into multilingual translation models of low-resource language pairs. One of the most common approaches for this objective is bridging, in which translation between two disparate languages  $A$

and  $C$  with little or no parallel training data is accomplished by introducing one or more intermediary languages  $B_i$  with sufficient parallel training data. Our results can be used to inform which language pairs are likely to be helpful if selected as intermediaries, based on similarities in lexical semantics. For many of these languages, POS parsers may not be readily available. Hence, we are also investigating transfer / multitask learning approaches to sharing available POS information across several languages.

## Numeracy-preserving Word Embeddings

Word embedding models are typically able to capture the semantics of words via the distributional hypothesis, but fail to capture the numerical properties of numbers that appear in a text. This leads to problems with numerical reasoning involving tasks such as question answering. We propose a new methodology to assign and learn embeddings for numbers. Our approach creates Deterministic, Independent-of-Corpus Embeddings (referred to as DICE) for numbers, such that their cosine similarity reflects the actual distance on the number line. DICE outperforms a wide range of pre-trained word embedding models across multiple examples of two tasks: (i) evaluating the ability to capture numeration and magnitude; and (ii) performing list maximum, decoding, and addition. We further explore the utility of these embeddings in downstream applications by initializing numbers with our approach for the task of magnitude prediction. We also introduce a regularization approach to learning model-based embeddings of numbers in a contextual setting.

## 4.1 Introduction

Word embeddings capture semantic relationships between words by operationalizing the distributional hypothesis [Har54, Fir57]. They can be learned either non-contextually [MSC<sup>+</sup>13, PSM14, BGJM17] or contextually [DCLT18, PNI<sup>+</sup>18]. Non-contextual embeddings have worked well on various language understanding and semantic tasks [RHW<sup>+</sup>88, MCCD13, MSC<sup>+</sup>13]. More recently, they have also been used as pre-trained word embeddings to aid more sophisticated contextual models for solving rigorous natural language processing (NLP) problems, including translation, paraphrasing, and sentence-similarity tasks [KZS<sup>+</sup>15, WBGL15, SCS<sup>+</sup>19].

While word embeddings effectively capture semantic relationships between *words*, they are less effective at capturing numeric properties associated with *numbers*. There are a variety of tests proposed to measure the efficiency of number embeddings. For instance, [NRRH19] shows that GloVe [PSM14], word2vec [MSC<sup>+</sup>13], and fastText [JGBM16, BGJM17] fail to capture *numeration* and *magnitude* properties of a number. Numeration is the property of associating numbers with their corresponding word representations (“3” and “three”) while magnitude represents a number’s actual value ( $3 < 4$ ). Further, [WWL<sup>+</sup>19a] proposes several tests for analyzing numerical reasoning of number embeddings that include list maximum, decoding, and addition.

In this paper, we experimentally demonstrate that if the cosine similarity between word embeddings of two numbers reflects their actual distance on the number line, the resultant word embeddings are useful in downstream tasks. We first demonstrate how *Deterministic, Independent-of-Corpus Embeddings* (DICE) can be constructed such that they almost perfectly capture properties of numeration and magnitude. These non-contextual embeddings

also perform well on related tests for numeracy [WWL<sup>+</sup>19a].

To demonstrate the efficacy of DICE for downstream tasks, we explore its utility in two experiments. First, we design a DICE embedding initialized Bi-LSTM network to classify the magnitude of masked numbers in the 600K dataset [CHTC19]. Second, given the popularity of modern contextual model-based embeddings, we devise a regularization procedure that emulates the hypothesis proposed by DICE and can be employed in any task-based fine-tuning process. We demonstrate that adding such regularization helps the model internalize notions of numeracy while learning task-based contextual embeddings for the numbers present in the text. We find promising results in a numerical reasoning task that involves numerical question answering based on a sub-split of the popular SQuAD dataset [RZLL16].

Our contribution can be summarized as follows:

- We propose a deterministic technique to learn numerical embeddings. DICE embeddings are learned independently of corpus and effectively capture properties of numeracy.
- We prove experimentally that the resultant embeddings learned using the above methods improve a model’s ability to reason about numbers in a variety of tasks, including numeration, magnitude, list maximum, decoding, and addition.
- We also demonstrate that properties of DICE can be adapted to contextual models, like BERT [DCLT18], through a novel regularization technique for solving tasks involving numerical reasoning.

## 4.2 Related Work

The major research lines in this area have been dedicated to (i) devising probing tests and curating resources to evaluate the numerical reasoning abilities of pre-trained embeddings, and (ii) proposing new models that learn these properties.

[NRRH19] surveyed a number of non-contextual word embedding models and highlighted the failure of those models in capturing two essential properties of numbers – *numeration* and *magnitude*. [CHTC19] created a novel dataset named Numeracy-600k, a collection of approximately 600,000 sentences from market comments with a diverse set of numbers representing age, height, weight, year, etc. The authors use neural network models, including a GRU, BiGRU, CRNN, CNN-capsule, GRU-capsule, and BiGRU-capsule, to classify the magnitude of each number. [WWL<sup>+</sup>19a] compares and contrasts the numerical reasoning ability of a variety of non-contextual as well as contextual embedding models. The authors also proposed three tests – list maximum, decoding, and addition – to judge the numerical reasoning ability of embeddings of numerals. They infer that word embedding models that perform the best on these three tests have captured the numerical properties of numbers well. Therefore, we consider these proposed tests in our evaluation. [SR18] used a variety of models to distinguish numbers from words, and demonstrated that this ability reduces model perplexity with neural machine translation. [WGY18] found that neural networks are capable of reasoning numbers with explicit supervision.

Numerically Augmented QANet (NAQANet) [DWD<sup>+</sup>19] was built by adding an output layer on top of QANet [YDL<sup>+</sup>18] to predict answers based on addition and subtraction over numbers in the DROP dataset. Our work, in contrast, offers a simple methodology that can be added to any model as a regularization technique. Our work is more similar

to [JNG<sup>+</sup>19], where the embedding of a number is learned as a simple weighted average of its prototype embeddings. Such embeddings are used in tasks like word similarity, and sequence labeling and have been proven to be effective.

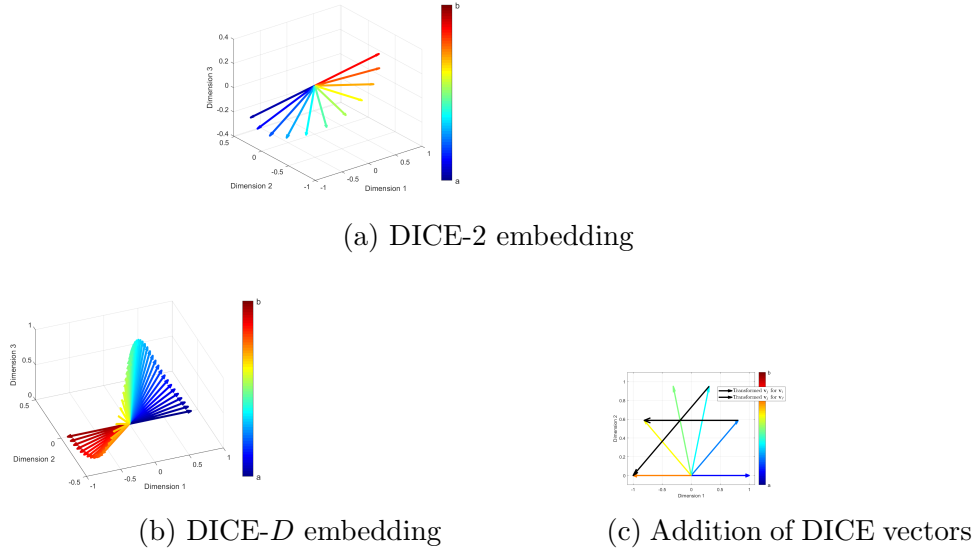


FIGURE 4.1: Proposed DICE embeddings. Vectors are colored according to numeral magnitude. Note that the addition of two numbers in this embedding is performed by a shift, scaling, and rotation. Scaling depends only on the vector being added, as illustrated in sub-figure (c) in which the two black lines, corresponding to identical  $\mathbf{e}_j$ , have the same length.

### 4.3 Methods

To overcome NLP models inefficiency in dealing with numbers, we consider our method DICE to form embeddings. To begin, we embed numerals and word forms of numbers as vectors  $\mathbf{e}_i \in \mathbb{R}^D$ , where  $i$  indexes numerals identified within a corpus. We first preprocess by parsing the corpora associated with each of our tasks (described below) for numbers in numeral and word forms to populate a number vocabulary. Then, the dimensionality of the embeddings required for that task is fixed. We explicitly associate the embeddings of a numeral and word forms of numbers to have the same embedding.

### 4.3.1 DICE embeddings

In designing embeddings that capture the aforementioned properties of numeration and magnitude, we consider a deterministic, handcrafted approach (depicted in Figures 4.1a and 4.1b). This method relies on the fact that tests for both numeration and magnitude are concerned with the correspondence in the similarity between numbers in token space and numbers in embedding space. In token space, two numbers  $x, y \in \mathbb{R}$ , in numeral or word form (with the latter being mapped to its corresponding numeral form for comparison), can be compared using absolute difference, *i.e.*:

$$d_n(x, y) = |x - y| \tag{4.1}$$

The absolute value ensures that two numbers are treated as equally distant regardless of whether  $x \geq y$  or  $y \geq x$ . On the other hand, two embeddings  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$  are typically compared via cosine similarity, given by:

$$s_e(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \cos(\theta) \tag{4.2}$$

$$d_e(\mathbf{x}, \mathbf{y}) = 1 - \cos(\theta) \tag{4.3}$$

where  $\theta$  is the angle between  $\mathbf{x}$  and  $\mathbf{y}$  and  $d_e(\mathbf{x}, \mathbf{y})$  is their cosine distance. Normalization by the vector lengths ensures that the metric is independent of the lengths of the two vectors.

Note that numerals are compared in terms of distance while their embeddings are compared by similarity. As cosine distance increases, the angle between  $\mathbf{x}$  and  $\mathbf{y}$  increases monotonically. A distance of zero is achieved when  $\mathbf{x}$  and  $\mathbf{y}$  are oriented in the same direction. When  $\mathbf{x} \perp \mathbf{y}$ , the cosine distance is 1; and when  $\mathbf{x}$  and  $\mathbf{y}$  are antiparallel, cosine

distance is 2.

We seek a mapping  $(x, y) \mapsto (\mathbf{x}, \mathbf{y})$  such that  $d_e$  monotonically increases as  $d_n$  increases. We first bound the range of numbers for which we wish to compute embeddings by  $[a, b] \subset \mathbb{R}$  and, without loss of generality, restrict  $\mathbf{x}$  and  $\mathbf{y}$  to be of unit length (*i.e.*,  $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ ). Since the cosine function decreases monotonically between 0 and  $\pi$ , we can simply employ a linear mapping to map distances  $s_n \in [0, |a - b|]$  to angles  $\theta \in [0, \pi]$ :

$$\theta(s_n) = \frac{s_n}{|a - b|} \pi \tag{4.4}$$

This mapping achieves the desired direct relationship between  $s_n$  and  $d_e$ . Since there are infinitely many choices for  $\mathbf{x}$  and  $\mathbf{y}$  with angle  $\theta$ , we simply fix the direction of the vector corresponding to the numeral  $a$ . Numbers that fall outside  $[a, b]$  are mapped to a random angle in  $[-\pi, \pi]$ . In the corpora we considered,  $a$  and  $b$  are chosen such that numbers outside  $[a, b]$  represent a small fraction of the total set of numbers (approximately 2%).

We employ this mapping to generate numeral embeddings in  $\mathbb{R}^D$ . Figure 4.1a shows deterministic, independent-of-corpus embeddings of rank 2 (DICE-2). In this approach we represent angles as vectors in  $\mathbb{R}^2$  using the polar-to-Cartesian coordinate transformation:

$$[r, \theta] \mapsto [x_1, x_2] = [r \cos(\theta), r \sin(\theta)] =: \mathbf{v} \tag{4.5}$$

where we choose  $r = 1$  without loss of generality. We then sample a random matrix  $\mathbf{M} \in \mathbb{R}^{D \times D}$  where  $D \geq 2$  and  $m_{ij} \sim \mathcal{N}(0, 1)$  and perform a QR decomposition on  $\mathbf{M}$  to obtain a matrix  $\mathbf{Q}$  whose columns  $\mathbf{q}_i, i = 1, \dots, D$  constitute an orthonormal basis for  $\mathbb{R}^D$ . The DICE-2 embedding  $\mathbf{e} \in \mathbb{R}^D$  of each numeral is then given by  $\mathbf{e} = \mathbf{Q}_{1:2} \mathbf{v}$ , where the subscript on  $\mathbf{Q}$  indicates taking the first two columns of  $\mathbf{Q}$ .

In Figure 4.1b we consider DICE- $D$ , in which we generate vectors in  $\mathbb{R}^D$  by applying a

Table 4.1: Performance (% accuracy) on numeracy tests.

Model	OVA	SC	BC
Random	0.04	48.92	49.34
Glove 6B-200D	15.88	62.21	83.94
Glove 6B-300D	18.41	62.92	83.98
Glove-840B-300D	5.18	55.58	91.86
FastText-Wiki	13.94	59.96	96.15
FastText-CC	7.83	53.89	85.40
Skip-gram-5	8.85	55.40	96.42
Skip-gram-Dep	3.32	51.99	94.60
DICE- $D$ (ours)	<b>95.63</b>	<b>99.66</b>	<b>99.64</b>

polar-to-Cartesian transformation in  $D$  dimensions [Blu60]:

$$v_d = \begin{cases} [\sin(\theta)]^{d-1} \cos(\theta), & 1 \leq d < D \\ [\sin(\theta)]^D, & d = D \end{cases} \quad (4.6)$$

where the subscripts indicate the coordinate in  $\mathbf{v}$ . We again apply a QR-decomposition on a random matrix  $\mathbf{M}$  generated as above, except here we project  $\mathbf{v}$  using all  $D$  basis vectors. This allows for a random rotation of the embeddings to avoid bias due to choosing  $e_{a1} = 1, e_{ai} = 0 \forall i \neq 1$ . We employ DICE-D embeddings throughout this paper as word embeddings are practically not 2 dimensional.

## 4.4 Experiments

To observe the numerical properties of DICE, we consider two tasks: **Task 1** deals with the *numeration* ( $NUM$ ) and *magnitude* ( $MAG$ ) properties as proposed by [NRRH19]; **Task 2** performs *list maximum*, *decoding*, and *addition* as proposed by [WWL<sup>+</sup>19a]. We then experiment on two additional tasks to demonstrate the applications of DICE.

Table 4.2: Experimental results on list maximum, decoding, and addition using the DICE- $D$  method.

<i>Integer range</i>	List maximum (accuracy)			Decoding (RMSE)			Addition (RMSE)		
	[0, 99]	[0, 999]	[0, 9999]	[0, 99]	[0, 999]	[0, 9999]	[0, 99]	[0, 999]	[0, 9999]
Random vectors	0.16	0.23	0.21	29.86	292.88	2882.62	42.03	410.33	4389.39
Untrained CNN	0.97	0.87	0.84	2.64	9.67	44.40	1.41	14.43	69.14
Untrained LSTM	0.70	0.66	0.55	7.61	46.5	210.34	5.11	45.69	510.19
Value embedding	<b>0.99</b>	0.88	0.68	1.20	11.23	275.50	<b>0.30</b>	15.98	654.33
<b><i>Pretrained</i></b>									
Word2Vec	0.90	0.78	0.71	2.34	18.77	333.47	0.75	21.23	210.07
GloVE	0.90	0.78	0.72	2.23	13.77	174.21	0.80	16.51	180.31
ELMo	0.98	0.88	0.76	2.35	13.48	62.20	0.94	15.50	45.71
BERT	0.95	0.62	0.52	3.21	29.00	431.78	4.56	67.81	454.78
<b><i>Learned</i></b>									
Char-CNN	0.97	<b>0.93</b>	0.88	2.50	4.92	11.57	1.19	7.75	<b>15.09</b>
Char-LSTM	0.98	0.92	0.76	2.55	8.65	18.33	1.21	15.11	25.37
<b><i>DROP-trained</i></b>									
NAQANet	0.91	0.81	0.72	2.99	14.19	62.17	1.11	11.33	90.01
NAQANet (w/out GloVe)	0.88	0.90	0.82	2.87	5.34	35.39	1.45	9.91	60.70
<b><i>Ours</i></b>									
DICE- $D$	0.98	0.87	<b>0.96</b>	<b>0.43</b>	<b>0.83</b>	<b>3.16</b>	0.75	<b>2.79</b>	29.95

#### 4.4.1 Task 1: Exploring Numeracy

In this task, proposed by [NRRH19], there are three tests for examining each property of numeration (NUM, 3 = “three”) and magnitude (MAG, 3 < 4). For each of these tests, target numbers in its word or numeral form are evaluated against other numbers as follows:

- *One-vs-All (OVA)*: The distance between the embedding vector of the target and its nearest neighbor should be smaller than the distance between the target and any other numeral in the data.
- *Strict Contrastive (SC)*: The distance of the embedding vector of the target from its nearest neighbor should be smaller than its second nearest neighbor numeral.
- *Broad Contrastive (BC)*: The distance of the embedding vector of the target numeral from its nearest neighbor should be smaller than its furthest neighbor.

*Training Details.* We use the Gigaword corpus obtained from the Linguistic Data Consortium to populate the list of numbers from the dataset. Parsing was performed using the `text2digits`<sup>1</sup> Python module. As done by [NRRH19], we employ  $D = 300$  for the DICE- $D$  embeddings. Embeddings of numerals are assigned using the principle explained in Section 4.3.1, while the embedding of words that denote numbers (word form) simply points to the embedding of that numeral itself. We then perform the six tests (OVA-NUM / OVA-MAG, SC-NUM / SC-MAG, BC-NUM / BC-MAG) on 130 combinations of numbers for NUM and 31,860 combinations of numbers for MAG.

*Evaluation.* Following [NRRH19], we use accuracy to measure the efficiency of the embeddings. These tests require the fulfillment of certain clauses which are defined in [NRRH19].

*Results.* Table 4.1 shows comparisons of the performance of embeddings created by each of the DICE methods on the MAG tests. Compared to the baselines, both DICE methods outperform all commonly employed non-contextual word embedding models in OVA, SC, and BC tests. This is attributed to the cosine distance property addressed in the DICE embeddings. Specifically, because the magnitude of the number is linearly related to its angle, sweeping through numbers in order guarantees an increase in angle along each axis. Numbers that are close to each other in magnitude are rotated further but in proportion to their magnitude. Thus, small and large numbers are ensured to lie near other small and large numbers, respectively, in terms of cosine distance.

On the NUM tests, DICE achieves perfect accuracy. The primary reason DICE embeddings perform so well on numeracy tasks is that the preprocessing steps taken allow us to parse a corpus for word forms of numbers and explicitly set matching embeddings

<sup>1</sup> <https://pypi.org/project/text2digits/>

for both word and numeral forms of numbers. Each of these embeddings is guaranteed to be unique since a number’s embedding is based on its magnitude, *i.e.*, the larger the magnitude, the greater the angle of the embedding, with a maximum angle of  $\pi$ . This ensures that the numeral form of a number is always able to correctly identify its word form among all word forms in the corpus as that with the smallest cosine distance (which equals zero). Performance on OVA-NUM is a lower bound on the performance of SC-NUM and BC-NUM, so those tests are guaranteed to pass under our approach.

#### *4.4.2 Task 2: List Maximum, Decoding, and Addition*

This task considers the operations proposed by [WWL<sup>+</sup>19a] – list maximum, decoding, and addition. List maximum deals with the task of predicting the maximum number given the embedding of five different numbers. Decoding deals with regressing the value of a number given its embedding. An additional task involves predicting the sum of two numbers given their embeddings.

*Training Details.* The list-maximum test presents to a Bi-LSTM neural network a set of five numbers of the same magnitude, and the network is trained to report the index of the maximum number. In the decoding test, a linear model and a feed-forward network are each trained to output the numeral corresponding to the word form of a number based on its embedding. Finally, in the addition test, a feed-forward network is trained to take in the embeddings of two numbers as its input and report the sum of the two numbers as its output. Each test is performed on three ranges of integers  $[0, 99]$ ,  $[0, 999]$ , and  $[0, 9999]$ , with an 80/20 split of training and testing data sampled randomly. The neural network is fed with the embedding of numbers; the task is either classification (in the case of

list maximum) or prediction of a continuous number (in case of addition and decoding). We replicate the exact experimental conditions and perform the three tests with DICE embeddings. For the sake of consistency with the tests proposed by [WWL<sup>+</sup>19a], we also only deal with positive in this experiment.

*Evaluation.* List maximum again uses accuracy as its metric while decoding and addition use root mean squared error (RMSE), since predictions are continuous.

*Results.* Given the strong performance of the DICE-*D* method on the NUM and MAG tests, we next consider its performance on tasks involving neural network models. In their empirical study, [WWL<sup>+</sup>19a] compared a wide range of models that included a random baseline; character level models such as a character-CNN and character-LSTM, which were both untrained and trained; a so-called value embedding model in which numbers are embedded as their scalar value; traditional non-contextual word embedding models including Word2Vec and GloVe; contextual word embedding models including ELMo and BERT; and the Numerically Aware Question Answering (NAQA) Network, a strong numerical reasoning model proposed on the Discrete Reasoning over Paragraphs (DROP) dataset.

We compare the performance of our DICE-*D* embedding to that of the other models on each of the three tasks proposed by [WWL<sup>+</sup>19a]. Results are presented in Table 4.2. We find that our DICE embedding exceeds the performance of more sophisticated models by large margins in all but four cases. In two of those four, our model fell short by only a few percentage points. We attribute the success of the DICE-*D* approach to the fact that the model is, by design, engineered to handle numeracy. Just as the value embedding model – which proved to be reasonably successful in all three tasks across a wide range of

numbers – captures numeracy through the magnitude of embeddings, our model captures numeracy through the angle corresponding to the embeddings.

The value embedding model, however, breaks down as the range of the processed numbers grows. This is likely because, as demonstrated by [THR<sup>+</sup>18], networks trained on numeracy tasks typically struggle to learn identity mapping. We reason that our model outperforms the value embedding model because the network learns to associate features between the set of inputs such that the input vectors can be scaled, rotated, and translated in  $D$  dimensions to achieve the desired goal.

More precisely, for a neural network to learn addition, numbers must be embedded such that their vector embeddings can be consistently shifted, rotated, and scaled to yield the embedding of another number (see Figure 4.1c). The choice of embedding is essential as it may be impractical for a network to learn a transformation for all embeddings that obeys this property (without memorization).

DICE is quite similar to the value embedding system, which directly encodes a number’s value in its embeddings. However, DICE performs better due to its compatibility with neural networks, whose layers are better suited for learning rotations and scaling than identity mappings.

Finally, both the value embedding models for a small number range and the character level models remain somewhat competitive, suggesting again that exploring a digit-by-digit embedding of numerals may provide a means of improving our model further.

## 4.5 Applications of DICE

### 4.5.1 Magnitude Classification

We examine the importance of good initialization for number embedding vectors [KB17b], particularly for better contextual understanding. In particular, we experiment on the magnitude classification task, which requires the prediction of magnitudes for masked numbers. The task is based on the 600K dataset proposed by [CHTC19], which requires classification into one of seven categories corresponding to powers of 10 in  $\{0, 1, 2, 3, 4, 5, 6\}$ .

*Training Details.* We use a bi-LSTM [HS97] with soft attention [CBS<sup>+</sup>15] to classify the magnitude of masked numbers. Numerals are initialized with corresponding DICE embeddings, and the target number is masked by substituting a random vector. Each token  $x_n$  in a sequence of length  $N$  is associated with a forward and backward LSTM cell. The hidden state  $\mathbf{h}_n$  of each token is given by the sum of the hidden states of the forward and backward cells:  $\mathbf{h}_n = \overleftarrow{\mathbf{h}}_n + \overrightarrow{\mathbf{h}}_n$ . To generate a context vector  $\mathbf{c}$  for the entire sentence, we compute attention scores  $\alpha_n$  by taking the inner product of each hidden state  $\mathbf{h}_n$  with a learned weight vector  $\mathbf{w}$ . The resulting scores are passed through a softmax function, and the weights are used to form a convex combination of the  $\mathbf{h}_n$  that represents the context  $\mathbf{c}$  of the sentence. Logits are obtained by taking the inner product of  $\mathbf{c}$  with trained embeddings for each of the seven categories, and cross-entropy loss is minimized. More details on training can be found in Section 4.6.

*Evaluation.* Following [CHTC19], we use micro and macro F1 scores for classifying the magnitude of a number.

Table 4.3: Performance (%) on classifying number magnitude on the Numeracy-600k dataset.

<b>Model</b>	<b>Micro-F1</b>	<b>Macro-F1</b>
LR	62.49	30.81
CNN	69.27	35.96
GRU	70.92	38.43
BiGRU	71.49	39.94
CRNN	69.50	36.15
CNN-capsule	63.11	29.41
GRU-capsule	70.73	33.57
BiGRU-capsule	71.49	34.18
BiLSTM with DICE	<b>75.56</b>	<b>46.80</b>

Table 4.4: Performance (%) of BiLSTM-attention with DICE model on the Numeracy-600k dataset by varying the embedding dimensions of input tokens.

<b>Embedding Size</b>	<b>Micro-F1</b>	<b>Macro-F1</b>
32	74.63	45.92
64	74.90	45.99
128	75.55	46.36
256	<b>75.56</b>	45.56
512	74.14	<b>46.80</b>

*Results.* Table 4.3 shows significant improvements in the F1 score achieved by the model. To investigate the effects of dimensions of the embedding and hidden vectors within the LSTM cells on the performance of the BiLSTM-attention model, we perform ablation experiments. We vary the embedding size of tokens while keeping other hyper-parameters constant, and observe the results on Tables 4.4. From Table 4.4 the BiLSTM with DICE model achieves the best micro-F1 score when the embedding dimension is 256. However, the macro-F1 score peaks when the embedding dimension is 512.

These results suggest that while DICE embeddings yield superior performance in non-contextual numerical tasks, such as computing the maximum and performing basic mathematical operations, data agnostic embeddings such as DICE may not be ideal for textual

reasoning tasks in which words surrounding a number provide important information regarding the magnitude of the number. Hence, we introduce a model-based regularization method that utilizes the DICE principles to learn number embeddings in 4.5.2.

#### 4.5.2 Model-Based Numeracy Embeddings

In the previous section, we demonstrated how DICE could be explicitly incorporated for numbers in the text. Here, we propose a methodology that helps models implicitly internalize the properties of DICE. Our approach involves a regularization method (an auxiliary loss) that can be adopted in the fine-tuning of any contextual NLP model, such as BERT. Auxiliary losses have been shown to work well for a variety of NLP downstream tasks [SCS<sup>+</sup>19].

During the task-specific training of any model, the proposed auxiliary loss  $\mathcal{L}_{num}$  can be applied to the input embeddings of numbers available in a minibatch. For any two contextual numerical embeddings  $\mathbf{x}, \mathbf{y}$  obtained from the final hidden layer of the model, the  $\mathcal{L}_{num}$  loss for the pair of numbers  $(x, y)$  is calculated as:

$$\mathcal{L}_{num} = 2 \frac{|x - y|}{|x| + |y|} - d_{\cos}(\mathbf{x}, \mathbf{y}) \quad (4.7)$$

*Training Details.* To evaluate the proposed  $\mathcal{L}_{num}$ , we test the regularization on the task of question answering (QA) involving numerical answers. In particular, we take the popular Stanford Question Answering Dataset (SQuAD 1.1) [RZLL16] dataset and create sub-splits (ranges from [1, 30000]) where the (i) training QA pairs have answers strictly containing numerical digits (Sub-split 1, less than 10K examples), and (ii) training QA pairs have answers containing a number as one of their tokens, for e.g. “10 apples” (Sub-split 2,

<p><b>A) Context</b></p> <p>According to the same statistics, the average age of people living in Newcastle is 37.8 (the national average being 38.6). Many people in the city have Scottish or Irish ancestors. There is a strong presence of Border Reiver surnames, such as Armstrong, Charlton, Elliot, Johnstone, Kerr, Hall, Nixon, Little and Robson. There are also small but significant Chinese, Jewish and Eastern European (Polish, Czech Roma) populations. There are also estimated to be between 500 and 2,000 Bolivians in Newcastle, forming up to 1% of the population—the largest such percentage of any UK city.</p>	<p><b>Question</b></p> <p>What is the smallest number of Bolivians it's estimated live in Newcastle?</p> <p><b>Answer</b></p> <p>Ground truth: 500</p> <p>BERT : between 500 and 2,000</p> <p>BERT + <math>\mathcal{L}_{num}</math> : 500</p>	<p><b>B) Context</b></p> <p>Although the reciprocating steam engine ... use, various companies ... alternative to internal combustion engines. The company Energiprojekt AB in Sweden ... the power of steam. The efficiency ... steam engine reaches some 27-30% on high-pressure engines. It is a single-step, 5-cylinder engine (no compound) with superheated steam and consumes approx. 4 kg (8.8 lb) of steam per kWh.</p>	<p><b>Question</b></p> <p>How many cylinders does the Energiprojekt AB engine have?</p> <p><b>Answer</b></p> <p>Ground truth: 5</p> <p>BERT : 27 - 30 % on high - pressure engines . it is a single - step , 5</p> <p>BERT + <math>\mathcal{L}_{num}</math> : 5</p>
--	---	--	---

FIGURE 4.2: Qualitative examples where BERT +  $\mathcal{L}_{num}$  performed better than BERT-base

slightly more than 10K examples). We create these splits to evaluate the BERT model’s reasoning involving numbers to pick these answers. We choose BERT-base-uncased as a baseline model and train it on both datasets. Within each batch, we calculate  $\mathcal{L}_{num}$  by randomly sampling a pair of numbers  $x, y$  from the available numbers in the contexts. The corresponding embeddings of the numbers are  $\mathbf{x}$  and  $\mathbf{y}$ , which are extracted from the last hidden layer of the BERT model. We then enforce the distance of embeddings to match the difference between number values by  $\mathcal{L}_{num}$ . The scores are reported on the development set (less than 1000 examples) as the test set cannot be pruned for our purpose. The assumption here is that the BERT model needs to perform numerical reasoning to come up with answers for these particular kinds of QA pairs. The models were trained on Nvidia Tesla P100 GPU. More details on choosing hyper-parameter for BERT +  $\mathcal{L}_{num}$  are discussed in Section 4.7.

*Evaluation.* Exact Match is a binary measure (*i.e.*, true/false) of whether the predicted output matches the ground truth answer exactly. This evaluation is performed after the string normalization (uncased, articles removed, etc.). F1 is the harmonic mean of precision and recall.

*Results.* Results in Table 4.5 show that the BERT model with numeracy objective achieves an improvement of 0.48 F1 points when the answers are purely numerical digits. When the BERT model is trained on QA pairs with answers containing at least a number with several words and evaluated on pairs with answers containing only numbers, we see an improvement of 1.12 F1 points over the baseline model.

The BERT-base model on the original SQuAD data was finetuned for 3 epochs owing to its complexity. However, we find that 1 epoch is sufficient to capture the complexity of the pruned SQuAD data. Table 4.5 shows BERT +  $\mathcal{L}_{num}$  consistently performs better than BERT-base across epochs.

Interestingly, BERT-base performs worse when finetuned with QA pairs containing a mix of words and numbers as answers (sub-split 2). This informs us that the baseline model learns to pick numbers better but fails to do as well when fine-tuned with a mix of words and numbers. In both cases, the evaluation set consists of pruned SQuAD dev set QA pairs with answers strictly containing numerical digits only. We find that BERT +  $\mathcal{L}_{num}$  gives the maximum improvement on sub-split 2 data highlighting the efficiency of our regularization technique to learn numerical embeddings.

Figure 4.2 shows some qualitative examples where the BERT +  $\mathcal{L}_{num}$  performs better than BERT-base (Sub-split 2). In this analysis, we found that the baseline model picks the whole sentence or paragraph involving the numerical value (Figure 4.2 B) as the answer. Our method picks numbers within the classification span (Figure 4.2 B) and sometimes helps the BERT model to accurately pick up correct numbers (Figure 4.2 A), contributing to exact match and F1. More such examples are shown in Section 4.8.

During our experiments, we observed the potential issue of weak signals from the loss

Table 4.5: F1 scores of BERT-base model on SQuAD 1.1 sub-splits (all scores are statistically significant with a variance of 0.01). **Sub-split 1**: both training and testing split contains only numerical answers; **Sub-split 2**: train split contains at least one number in the answer and testing split contains only numerical answers.

Model	Epoch	Sub-split 1		Sub-split 2	
		F1	Exact	F1	Exact
BERT	1	89.75	89.40	89.03	86.50
	2	90.71	90.66	90.32	88.09
	3	91.12	91.04	90.28	88.02
BERT + $\mathcal{L}_{num}$	1	<b>90.23</b>	90.16	<b>89.90</b>	87.26
	2	<b>91.05</b>	90.92	<b>90.49</b>	88.52
	3	<b>91.46</b>	91.29	<b>91.40</b>	89.15

when the availability of numerical pairs is sparse. In the future, our efforts would be to overcome this issue to ensure further gains.

#### 4.6 Training details for Magnitude Classification Experiment

The Bi-LSTM with attention model initialized with DICE embeddings was trained on the market comments data. The model was trained for a fixed number of 9 epochs. We found that the micro and macro F1 scores peaked for a certain epoch and then flattened out. We picked the best micro and macro pair the model obtained in that certain epoch.

#### 4.7 Hyperparameter for BERT + $\mathcal{L}_{num}$

Our model involves a regularization method (an auxiliary loss) that can be adopted in the fine-tuning of BERT. This loss was finetuned with a hyperparameter  $\lambda$  and added to the existing BERT classification loss for detecting the correct span. The hyperparameter search space is between 0, 1. We swept through the values manually within the search space and found that the best model that gave the maximum improvement in F1 scores had a

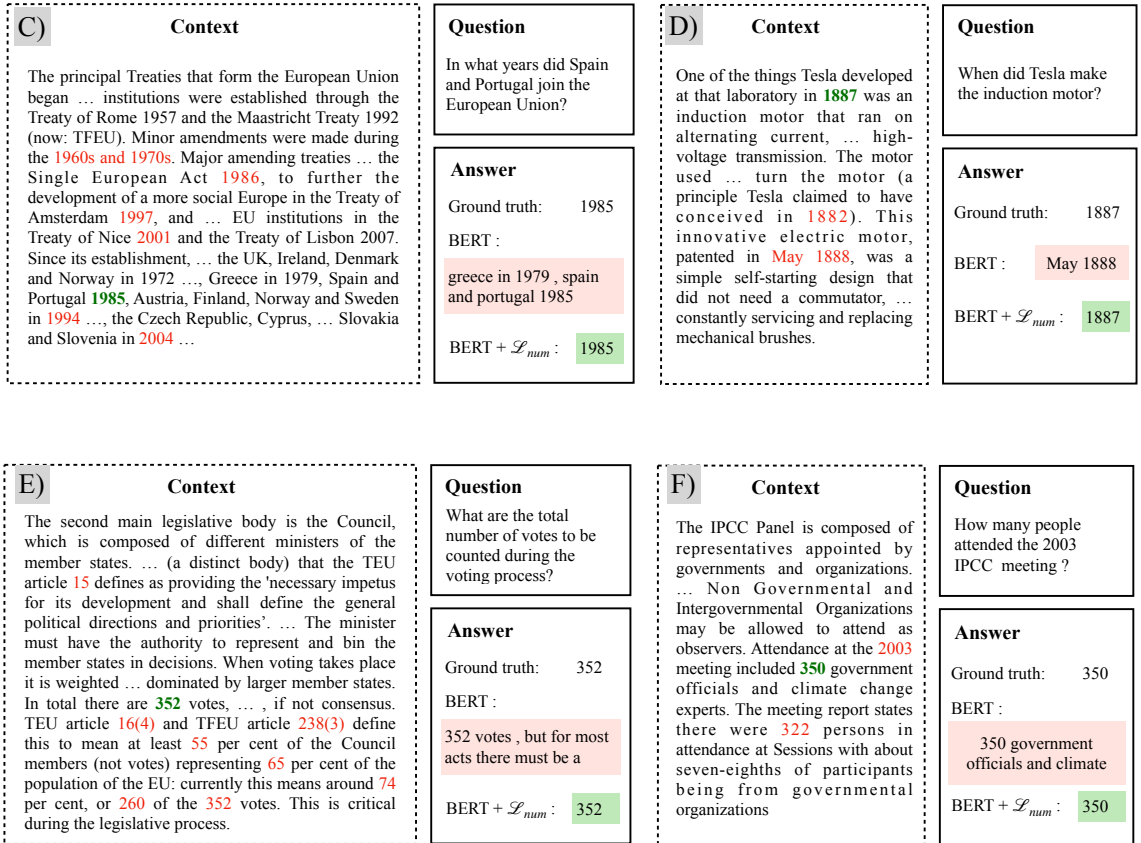


FIGURE 4.3: Qualitative examples where BERT +  $\mathcal{L}_{num}$  performed better than BERT base.

hyperparameter value of  $10^{-3}$ . The values were swept based on the observed performance.

The performance faded as the hyperparameter was set to a higher value (closer to 1).

#### 4.8 Examples for BERT vs. BERT + $\mathcal{L}_{num}$

Figure 4.3 provides additional samples where BERT +  $\mathcal{L}_{num}$  outperformed the baseline BERT model. Similar to previous observations, our regularized approach is able to pinpoint the correct number as opposed to selecting a substring via pattern matching.

## 4.9 Conclusion

In this work, we methodologically assign and learn embeddings for numbers to reflect their numerical properties. We validate our proposed approach with several experiments that test number embeddings. The tests that evaluate the numeral embeddings are fundamentally applicable to all real numbers. Finally, we introduced an approach to jointly learning embeddings of numbers and words that preserve numerical properties and evaluated them on a contextual word embedding-based model. In our future work, we would like to extend this idea to unseen numbers in vocabulary as a function of seen ones.

# Number Entity Recognition

Numbers are essential components of text, like any other word tokens, from which natural language processing (NLP) models are built and deployed. Though numbers are typically not accounted for distinctly in most NLP tasks, there is still an underlying amount of numeracy already exhibited by NLP models. In this work, we attempt to tap this potential of state-of-the-art NLP models and transfer their ability to boost performance in related tasks. Our proposed classification of numbers into entities helps NLP models perform well on several tasks, including a handcrafted Fill-In-The-Blank (FITB) task and on question answering, outperforming the BERT and RoBERTa baseline classification. It also helps multitask models to classify cross-numeracy through positive transfer while using shared parameters.

## 5.1 Introduction

Named entity recognition (NER) is the task of classifying nouns within a document into categories to which they belong [SDM03]. Deep learning approaches for NER have been suc-

Super bowl COUNT was an American football game to determine the champion of the NFL for the 2016 YEAR season.

The league announced on October 16 DATE, 2012 YEAR, that the two finalists were Sun Life’s stadium and Levi’s stadium.

FIGURE 5.1: Sequence classification of numbers

successful recently because of their ability to understand the underlying semantics of sentences and incorporate this knowledge into tagging. Deep architectures for NER [LBS<sup>+</sup>16] include recurrent neural networks and attention-based models including the Transformer [VSP<sup>+</sup>17].

While NER has been performed on a number of datasets, tagging types of numbers, or *number entities*, has not been a principal focus of NER. This is despite the fact that numbers are prevalent in most texts, comprising roughly 5% of tokens in a sentence [NRRH19]. Numbers can take on several categories, including years, ages, phone numbers, dates, etc. Figure 5.1 shows sample sentences in which numbers are classified into entity types. Distinguishing these different types of numbers can greatly enhance performance on downstream tasks.

In this paper, we present several methods for addressing *number entity recognition* (*NuER*). We begin by demonstrating that state-of-the-art NLP models are able to classify types of numbers that appear in text. We study this by utilizing BERT [DCLT18] to perform NuER on a custom version of the Stanford Question Answering Dataset (SQuAD) [RZLL16], pruned to contain roughly 10K sentences for which the answers consist of numbers, and annotated in-house for numbers of six different entity types (year, percentage, date, count, age, and size) [SSS<sup>+</sup>20]. We hereby refer to this as the SQuAD-Num dataset

and the BERT classifier used to predict entities as the NuER model. We discover that BERT model performs well at this task on most of the categories listed above. The success of this model is made possible by the highly contextualized embeddings (*e.g.*, the appearance of the name of a month, say March, next to a year, say 2003) inferred by BERT.

We then verify the effectiveness of BERT fine-tuned on SQuAD-Num to tag numbers that appear in an out-of-domain, human-annotated, 2K-sentence subset of the Numeracy 600K dataset [CHTC19], which we call *HA-Numeracy 2K*. We chose this dataset because of the abundance and diversity of numbers present. The subset was sampled in such a way as to maintain the distribution of the number magnitudes that are originally present. We find that contextual embeddings are again beneficial in this setting, as surrounding tokens in both the source and target domains tend to be related. After verifying the quality of the annotations on the small sample, we annotated the full Numeracy 600K dataset using this model, which we hereafter call *ENT-Numeracy 600K*. Table 5.1 shows the number of samples in each entity type in SQuAD-Num. The annotations were conducted individually by three annotators and their common agreements were marked as labels. These annotations form the basis of all the models that we develop. We share these annotations with the NLP community to further enhance research in this direction.

Next, we jointly train embeddings for both tokens and number entities on the SQuAD-Num dataset. Answers to questions involving numbers are predicted by combining together token embeddings with trainable number entity embeddings. We find that this addition enables BERT and RoBERTa models to better answer questions in SQuAD involving numbers. We later analyze the effectiveness of NuER through a handcrafted Fill-In-The-Blank

Table 5.1: Number of entities in our annotation for each type in SQuAD-Num.

Count	Size	Year	Percentage	Date	Age
1800	447	4355	291	418	82

(FITB) classification task. Our proposed NuER model helps the models classify numbers better when presented with entity-type information.

In summary, our contributions are as follows:

1. We develop a BERT-based model for NuER which is used to tag numbers that appear in the SQuAD-Num dataset and in the out-of-domain HA-Numeracy 2K.
2. We release datasets containing number entity annotations for the NLP community.
3. We observe the effectiveness of NuER through a handcrafted Fill-In-The-Blank (FITB) task to predict a masked number in a sentence.
4. We jointly train embeddings for tokens and number entities, and show that numerical question answering performance improves when both are fed into the model.

## 5.2 Related Work

Classification of numbers in plain text to entities has not been addressed methodologically and deserves considerable attention. [NRRH19] highlighted the importance of developing specialized embeddings for numbers such that basic properties including comparison and summation would be preserved. In addition, [SSS<sup>+</sup>20] designed a novel loss function that captured these properties and demonstrated improved performance on a variety of NLP tasks. Unlike their method which uses static embeddings, we propose to supplement the entity information of numbers in a contextual fashion to enable models to perform well

on a number of tasks. Existing works have also shown that NLP models are intrinsically capable of numerical reasoning. This potential can be tapped using simple modifications like those proposed by [AHL19], in which the BERT model was used to perform arithmetic operations. Like the above-mentioned works, we seek to leverage the potential of existing NLP models to classify numbers into different entity types and transfer such ability across related tasks to boost performance.

### 5.3 Preliminaries

For each corpus, we index sentences by  $n = 1, 2, \dots, N$ . Each token in a given vocabulary  $\mathcal{V}$  is assigned a one-hot vector embedding  $\mathbf{v} \in \{0, 1\}^{|\mathcal{V}|}$ . Tokens in a sequence are indexed by  $i = 1, 2, \dots, T_n$ , where  $T_n$  is the length of sequence  $n$  (after padding). During embedding lookup, tokens are mapped to embeddings  $\mathbf{e}_{ni} \in \mathbb{R}^d$  ( $d$  refers to the dimension of embedding). For all tasks, we use pre-trained models including BERT and RoBERTa to form latent representations of inputs; token embeddings are summed with position embeddings  $\mathbf{p}_{ni} \in \mathbb{R}^d$  and segment embeddings  $\mathbf{s}_{ni} \in \mathbb{R}^d$  prior to being fed into the Transformer encoder. The output consists of a classification token  $\mathbf{c}_n \in \mathbb{R}^d$  and a sequence of contextualized embeddings  $\mathbf{o}_{ni} \in \mathbb{R}^d$ , one for each token. For each model, we use an Adam optimizer with  $\eta = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-7}$  to minimize the empirical loss across samples in each batch.

### 5.4 Methods

#### 5.4.1 Number entity recognition

BERT, introduced by [DCL18], has been shown to exhibit state-of-the-art performance for NER. Token embeddings are passed through a Transformer encoder, generating contex-

tualized representations for each subword token, which are classified by the attention-based BERT classifier. We fine-tune the BERT model on the SQuAD-Num dataset. Both questions and context for each sequence are input to the NuER model, and the model outputs a sequence of tags denoting whether each token is one of the six types of numbers or “other” (seven total classes). The outputs of the NuER model for each element in the sequence, given by the  $\mathbf{o}_i$ , are mapped through a linear layer (we drop the  $n$  subscript for clarity), and the resultant scalars are fed into a softmax function:

$$\hat{y}_i = \frac{\exp(\mathbf{W}\mathbf{o}_i)}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{o}_i)} \quad (5.1)$$

where  $\mathbf{w} \in \mathbb{R}^d$ ,  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k, \dots, \mathbf{w}_K]^\top \in \mathbb{R}^{K \times d}$  are the parameters of the linear layer,  $k = 1, \dots, K$  indexes the entity labels, and the exp operator is applied elementwise. The model is trained with categorical cross-entropy loss:

$$\mathcal{L} = \sum_{i=1}^T \sum_{k=1}^K -y_{ik} \log(\hat{y}_{ik}) \quad (5.2)$$

where  $y_{ik} \in \{0, 1\}$  denotes whether the  $i$ th element belongs to class  $k$ . We divide SQuAD-Num into 75% training, 10% validation, and 15% testing. We fine-tune the NuER model for 20 epochs on SQuAD-Num and evaluate our model on the out-of-domain HA-Numeracy 2K.

#### 5.4.2 Jointly trained embeddings

We experiment with training embeddings for tokens jointly with number entities, referred as Jointly trained Embeddings (JEM) [WLW<sup>+</sup>18]. Specifically, we compare the performance of numerical question answering on SQuAD-Num, with and without entity embeddings. As input to the pre-trained model, we feed in pairs  $(\mathbf{v}_i, \mathbf{z}_i)$  of tokens  $\mathbf{v}_i$  and number entity

Table 5.2: Performance metrics for NuER on SQuAD-Num and ENT-Numeracy 600K for each of the six entity types.

		Year	Count	Percentage	Age	Size	Date	Total
SQuAD-Num	Precision	98.61	88.66	95.08	94.12	87.84	100.00	95.38
	Recall	98.99	95.91	96.67	88.89	84.42	100.00	97.24
	F1	98.8	92.15	95.87	91.43	86.09	100.00	96.3
ENT-Numeracy 600K	Precision	97.96	92.00	91.67	75.00	54.55	96.88	94.26
	Recall	96.48	81.01	95.65	7.14	30.00	85.94	84.14
	F1	97.21	86.16	93.62	13.04	38.71	91.09	88.91

Table 5.3: Few-shot transfer vs fine-tuning performance on NuER entity types.

		Year	Count
Few-shot	Precision	87.09	76.27
	Recall	97.32	89.59
	F1	91.92	82.39
Full finetuning	Precision	91.15	80.53
	Recall	98.49	95.33
	F1	94.68	87.31

labels  $\mathbf{z}_i$ .  $\mathbf{z}_i \in \{0, 1\}^{K+1}$  is a one-hot vector, where  $K = 6$  indicates the six entities of numbers in the corpus. During embedding lookup, the pairs of embeddings for each token are summed and fed as inputs:

$$\tilde{\mathbf{e}}_i = \mathbf{e}_i + \mathbf{h}_i \quad (5.3)$$

where  $\mathbf{h}_i$  denotes the trainable number entity embedding for token  $i$ . We compare the performance of this model after fine-tuning for two epochs against the baseline, which is only fed the token identities as input. As is standard for SQuAD, the model is trained with cross-entropy loss in order to predict the probability of any token in the context being the start/end of a span that answers the numerical question:

$$\mathcal{L} = \sum_{i=1}^T -y_i \log(\hat{y}_i) \quad (5.4)$$

where here,  $y_i \in \{0, 1\}$  identifies the start/end of a span, and  $\hat{y}_i$  is the corresponding probability predicted by our model.

### 5.4.3 FITB

In this subsection, we measure the effectiveness of NuER on classification tasks. We hand-craft a Fill-In-The-Blank (FITB) task of predicting a masked number in a given input sentence. This is challenging as there are several thousand unique numbers in the corpus to which a masked number could be classified. It measures the true capability of BERT to fill in a number given the context. Our model which we refer to as  $BERT_{NuER}$  consists of BERT tokenizer with six special tokens namely  $jsize\dot{}$ ,  $jage\dot{}$ ,  $jcount\dot{}$ ,  $jdate\dot{}$ ,  $jyear\dot{}$ , and  $jpercentage\dot{}$ . These special tokens are added next to the CLS as an indication to the model about their respective number entity types. On the classification side, a linear projection layer of size  $M \times D$  is created to learn about the six different entity types ( $M = 6$ ,  $D = 768$ ). These projections are then concatenated with the BERT hidden states of 768 dimensions to obtain the final set of dimensions  $(768 * 2) \times M$  which is projected again to a classification layer of  $V$  output vocabulary items. Here, the  $V$  output vocabulary items represent the unique numbers that are found in the corpus.

## 5.5 Results and Discussion

### 5.5.1 Number entity recognition

Table 5.2 shows the performance of our NuER model on each dataset. We find that the model performs well across all categories on SQuAD-Num and in most categories of ENT-Numeracy 600K. Performance drops are observed for the “Age” and “Size” categories, partially because there are not enough training instances of these two categories in SQuAD-

Table 5.4: Improvements on SQuAD-Num dataset using Jointly trained Embedding (JEM) technique with  $BERT_B$  (base),  $BERT_L$  (large), and  $RoBERTa_B$  (base). All numbers are tested for significance and have a standard deviation of 0.05.

Model	Version	Exact match	F1
$BERT_B$	Baseline	90.66	90.71
	JEM	<b>91.17</b>	<b>91.33</b>
$RoBERTa_B$	Baseline	95.83	95.88
	JEM	<b>96.84</b>	<b>96.95</b>
$BERT_L$	Baseline	93.19	93.28
	JEM	<b>94.83</b>	<b>95.01</b>

Num. For “Age,” precision is high while recall is low. This is a limitation of our work and could be because the numbers that are classified as “Age” can typically be confused with other entities including “Size,” suggesting that the model is only able to capture age accurately when it is quite obvious in the dataset. For “Size,” both precision and recall are low, perhaps because, unlike the other types of numbers, size may have many different units (*e.g.*, feet, meters, acres, *etc.*), which make it more difficult to associate with the numerical value. Table 5.3 shows that the classification performance when only few of the samples are used in fine-tuning is still comparable to the full fine-tuning performance on two classes. Though it works well for few classes, the F1 scores drop significantly for classes like Age and Size.

### *Training settings*

For Table 5.2 experiments, the learning rate was set to  $2 \times 10^{-5}$ . For few-shot used in Table 5.3, only the first 20 samples (using grid search, fewer samples to retain maximum accuracy compared to full finetuning) from Year and Count, while full fine-tuning uses all the samples.

<p><b>'dancing with the stars' : results for november 16, [MASK], season 11, week 9</b>            Answer: <b>2010</b>            Predictions: [2010, 2011, 2012, 2013, 2009, 13, 11, 10, 14, 12]</p>	<p><b>iphone [MASK] to begin production in July</b>            Answer: <b>5</b>            Predictions: [5, 4, 6, 7, 10, 3, 9, 2, 11, 8]</p>
<p><b>Toronto slides down on top Ontario investment cities 2010 – [MASK] list</b>            Answer: <b>2016</b>            Predictions: [2013, 2011, 2012, 13, 2014, 14, 2010, 11, 15, 10]</p>	<p><b>[MASK] plus degrees projected for this weekend 's tournament and night</b>            Answer: <b>100</b>            Predictions: [50, 100, 30, 90, 60, 40, 80, 21, 98, 75]</p>
<p><b>nelson garrett ' s antique show at the citadel september [MASK] in charleston , sc</b>            Answer: <b>2013</b>            Predictions: [13, 11, 25, 21, 10, 22, 15, 17, 18, 14]</p>	<p><b>national auto dealers association names honda ' s [MASK] cr - z ' car of the month !</b>            Answer: <b>2011</b>            Predictions: [2011, 2010, 2012, 2013, 2009, 2014, 2000, 2008, 2007, 2015]</p>

FIGURE 5.2: Qualitative comparison of predictions of numbers on the FITB task between the BERT base and  $BERT_{NuER}$ . The left column shows BERT base predictions, while the right column shows  $BERT_{NuER}$  predictions.  $BERT_{NuER}$  predictions align with a number entity more than BERT base.

### 5.5.2 Jointly trained embeddings

Table 5.4 compares the performance of our model trained on numerical question answering against the BERT baseline. Our model demonstrates gains in both the exact match and F1 scores, suggesting that the additional information imparted by knowledge of the type of numbers that are present in the question and context aids the model in localizing the number within the context. A variety of models including  $BERT_B$  (base),  $BERT_L$  (large) and  $RoBERTa_B$  (base) [DCLT18, LOG<sup>+</sup>19] are used to understand if these improvements are consistent. We found that in terms of overall performance,  $RoBERTa_B > BERT_L > BERT_B$ , with  $RoBERTa_B$  being the highest, while in terms of performance improvements,  $BERT_L > RoBERTa_B > BERT_B$  with  $BERT_L$  JEM obtaining an improvement of 1.64 exact match and 1.7 F1 points respectively.

Table 5.5: Predictive performance on the FITB task. All numbers are tested for significance and have a standard deviation of 0.01

	Top-1	Top-2	Top-5	Top-10
Baseline	36.88	52.51	71.57	79.84
Dist	624	1279.06	3275.22	8240.56
Ours	<b>37.69</b>	<b>53.9</b>	<b>73.1</b>	<b>81.47</b>
Dist	<b>554</b>	<b>1110</b>	<b>2829.73</b>	<b>6435.14</b>

*Training settings*

BERT for QA consists of a pre-trained model along with classification heads for the start and ends of spans. The model was trained for 2 epochs with a learning rate of  $2 \times 10^{-5}$ . During training, embeddings for numbers and their entities are learned in parallel.

*5.5.3 FITB*

Table 5.5 shows the performance improvements of  $BERT_{NuER}$  over the BERT base for FITB. The top- $k$  performance tells us that  $BERT_{NuER}$  is able to predict the correct answer within the top- $k$  entries. We see that the gap in performance improvement increases as  $k$  increases. This tells us that the enriched model is more accurate in its predictions. The dist metric quantifies the average absolute distance between pairs of actual values and the predicted values. This ensures the quality of numeral predictions. For example, as shown in the first column of Figure 5.2, when the ground truth is a Year, we expect all the other predictions to also be years (as opposed to Count or other classes). The dist metric calculates the spread of numbers while the top- $k$  calculates the accuracy itself. We see that the dist is significantly lower than the baseline model. Figure 5.2 shows that  $BERT_{NuER}$  generates top- $k$  predictions whose elements are more consistent with the desired class label.

### *Training settings*

For the FITB experiment, DistilBERT [SDCW19] with an Adam optimizer and a learning rate of  $2 \times 10^{-4}$  is used. The model is trained for 3 epochs with a batch size of 64. On the architectural side, adding a linear layer to the BERT hidden states yielded marginal improvements. But, concatenating the linear layer responsible for learning number entity onto the BERT hidden states followed by classification led to significant improvements.

## 5.6 Conclusions

We have defined number entities and find that this aids multiple NLP tasks, via joint training, and classification. We also introduce a dataset representing number entities and a methodology to annotate unseen numbers. In the classification objective, we find that the addition of entities provides a considerable boost in FITB evident through the quality of predictions. Through our datasets and methods, we hope to incite further research at the intersection of numeracy and NER.

# 6

## Learning Task Sampling Policy for Multitask Learning

It has been shown that training multi-task models with auxiliary tasks can improve the target tasks quality through cross-task transfer. However, the importance of each auxiliary task to the primary task is likely not known a priori. While the importance weights of auxiliary tasks can be manually tuned, it becomes practically infeasible with the number of tasks scaling up. To address this, we propose a search method that automatically assigns importance weights. We formulate it as a reinforcement learning problem and learn a task sampling schedule based on the evaluation accuracy of the multi-task model. Our empirical evaluation on XNLI and GLUE shows that our method outperforms uniform sampling and the corresponding single-task baseline.

### 6.1 Introduction

Multi-task learning [Car97] has been shown to improve the performance of multiple related tasks through cross-task knowledge transfer using just one model, which also drastically

improves parameter efficiency [HXTS16, KGS<sup>+</sup>17]. In the case where the objective is improving specific target tasks, the use of auxiliaries has demonstrated substantial benefits, for example, in sequence-to-sequence learning [LLS<sup>+</sup>15], and question answering [MKXS18].

In this paper, we follow this line and consider the goal of training a multi-task model to be the maximizing performance of one target task. While there are numerous efforts that attempted to increase the complexity of multi-task models to match the performance through architectural addition, not a lot of emphasis has been placed on harnessing the potential of a model through exploration of task sampling weights, as we view a weighted combination of different tasks as a sampling problem.

In our work, we formulate this exploration problem as learning a policy to assign task sampling weights differentially over time. For example, an optimal policy could be putting more weight on auxiliary tasks with rich data at the beginning of training and shifting to the target tasks near the end for finetuning. While manual hyper-parameter fine-tuning should help find a good policy, it becomes challenging as the number of combinations increases exponentially with respect to the number of tasks. To solve this problem, we propose a reinforcement-learning-based search method that automatically finds the optimal policy to maximize the target task accuracy. Different from previous dynamic re-weighting works [GPB19, WTN20], our policy is static and agnostic to model training artifacts such as which training examples are selected and network parameters, and thus the policy can also be viewed as a schedule. This property enables the reuse of a fixed sampling schedule once the search is done. When we need to re-train the model due to system change or minor underlying model change, it allows us to directly retrain without performing a policy search again.

We conduct experiments by targeting at improving one language’s performance in the XNLI [CLR<sup>+</sup>18] dataset or one task’s performance in the GLUE dataset. We empirically demonstrate that multi-task models trained with the proposed method outperform multi-task models learned with various rule-based sampling as well as corresponding single-task baselines. We also conduct necessary ablation studies to validate our approach. Our main contributions are as follows. First, we are the first work to formulate multi-task exploration as a static sampling policy/schedule learning problem to the best of our knowledge. Then, we propose a simple and effective policy optimization algorithm for learning task sampling policy.

## 6.2 Related Works

Existing works [CW08, LLS<sup>+</sup>15] show that, by doing proper architecture changes, we can utilize cross-task transfer of multi-task learning and have multi-task models outperform single-task baselines. However, for different multi-task settings, the optimal architectures may be different and difficult to know a priori. It leads us to prefer an automatic searching approach, e.g. RL-based architecture search methods [MZC<sup>+</sup>19] were proposed to learn how to share under different settings.

While there are many successes in searching architecture for multi-task learning, how to select tasks to train together is underexplored. The current approaches are somehow arbitrary. With a goal of optimizing the performance for one target task, some works [BS17, PSN<sup>+</sup>19, VWM<sup>+</sup>20] exhaustively explored the relatedness of tasks in a multi-task training setting by manually picking pair of tasks and found that the multi-task performance gains significantly. This becomes practically infeasible as the number of tasks grows to make the possible combinations of tasks exponential. Up-sampling strategy [JSL<sup>+</sup>17b, CLR<sup>+</sup>18],

such as uniform sampling that balances high-resource and low-resource tasks, is another dimension that has been explored. Finally, the sampling strategy can also change over time. Curriculum learning [PSN<sup>+</sup>19] can be used to choose to learn hard tasks first and easy tasks later. Domain adaption by continuous pre-training on the target domain [GMS<sup>+</sup>20] can be seen as a schedule that first pre-train on a general domain and then on a specific domain.

Instead of manually trying all the task combination options, reward-based learning methods have been proposed by [GPB19, WTN20] to solve the problem to maximize task transfer. However, such methods, which implicitly or explicitly depend on artifacts during training such as evaluation accuracy of the current timestamp or example-level weight, suffer from instability of rewards and randomness of state changes. As a result, it can be difficult to re-train to ensure to get good model quality.

Unlike previous works, we focus on learning a *static sampling schedule*, which means our schedule, once learned, should be fixed before any future model training. Future model training should not involve policy optimization. That makes it easy to re-train the models when the underlying system changes or share the sampling schedule with other teams who do not have policy-optimization expertise in an industrial setting.

## 6.3 Methods

In Sec. 6.3.1, we introduce our algorithms for learning task sampling policy. In Sec. 6.3.2, we discuss our exploration strategies.

### 6.3.1 Learning Task Sampling Policy

We consider the task sampling policy to be discrete and stochastic. At time step  $t$ , the policy outputs a distribution  $\pi(\cdot|s_t)$  over the actions ( $s_t$  is the state that the policy depends

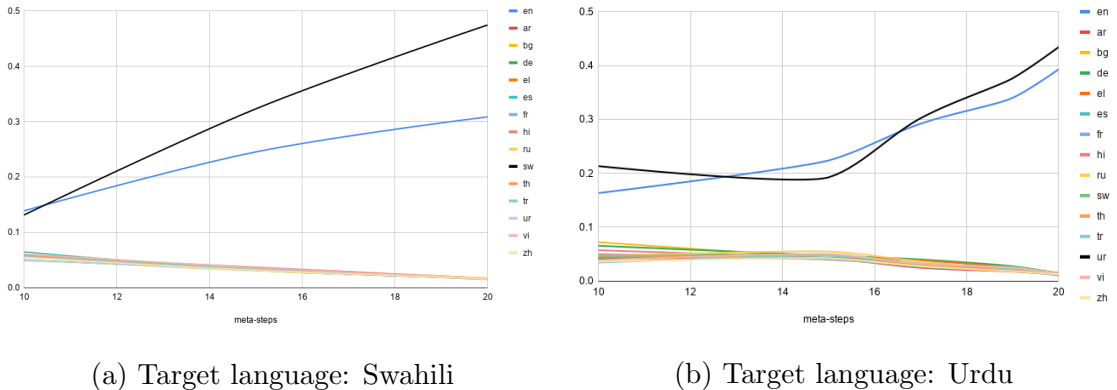


FIGURE 6.1: The sampling schedule learned for different target languages across meta-steps. The black line is the target language’s translate-train data and the blue line is English. We can see the learned sampling policy change as the target languages change.

on). Here we define an action to be a choice of the task to sample data from, and we sample a batch at each time step.

Our objective is to maximize the expected reward  $R$  which we define as the model’s evaluation accuracy on the target task:

$$\mathbb{E}_{\pi} \left[ \frac{1}{K} \sum_{k=0}^K \sum_{t=0}^T R(s_t^{(k)}, a_t^{(k)}) \right], \quad (6.1)$$

where  $K$  is the batch size, and  $k$  marks the  $k$ -th sample.

We parameterize the policy  $\pi$  with  $\theta$ , and learn it with REINFORCE policy gradient [SMS<sup>+</sup>99] (we ignore batch in notation):

$$\mathbb{E}_{\pi_{\theta}} \left[ \sum_{t=0}^T R(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) \right]. \quad (6.2)$$

Our goal is to learn a static sampling schedule that does not depend on complex dynamics of model parameter changes during training but only relies on timestamp, which

makes  $s_t$  only contain a timestamp. In practice, running an evaluation on the validation set to obtain a reward signal at each training step could be costly. Therefore, we define a meta-step to group  $N$  continuous training steps by creating a mapping  $M(t)$  that maps a step  $t$  to a meta-step, and only run evaluation at each meta-step.

In this work, our policy  $\pi_{\theta_{M(t)}}$  is simply parameterized as a vector of logits that defines a Boltzmann distribution  $\text{Softmax}(\theta_{M(t)})$  at each time step. The logits  $\theta_{M(t)}$  are trainable weights and uniformly initialized

We consider two different types of policy:

1. Time-variant policy: The policy at time step  $t$  is defined as  $\pi_{\theta_{M(t)}}$ . For all time steps in a meta-step, we use the same sampling policy.
2. Time-invariant policy: We only use one task sampling policy across all time steps within one run of model training, and thus  $\pi_{\theta_{M(t)}}$  reduces to  $\pi_{\theta}$ .

During training, the model is updated at each step using sampled examples by gradient descent. At each meta-step, we run evaluation on the full validation set to obtain accuracy, and use it as a reward to update  $\theta$  or  $\theta_{M(t)}$  through gradient ascent as in Eq. (6.2). This process ends until we reach the maximum number of training steps. While we can optionally repeat the training process with a randomly initialized model to further optimize the policy, practically we are able to find reasonably good sampling policy in one training run in all of our experiments.

After the searching is done, we re-train with the updated policy to generate the final optimized model.

Table 6.1: Performance improvements on the XNLI test dataset by our proposed methods. Our proposed methods, both the time-varying version and the time-invariant version, outperform various commonly used data re-weighting baselines.

Language	English	Target Language	Multi-task uniform	Multi-task time-variant	Multi-task time-invariant
Urdu	0.5693	0.6616	0.6618	0.6651	0.7032
Swahili	0.5094	0.6785	0.657	0.6877	0.7045
Thai	0.5386	0.6853	0.6847	0.7096	0.6853
Hindi	0.5903	0.6897	0.6811	0.6823	0.7127

### 6.3.2 Exploration Strategy

To facilitate exploration, the first strategy we consider is  $\epsilon$ -exploration [TP11], which refers to sampling actions from a uniform distribution with probability  $\epsilon$  and sampling from a policy with probability  $1 - \epsilon$  during training. The second strategy we consider is a heuristic-guided exploration that simply samples data from the target task.

We use a combination of these two. Let  $n$  be the number of tasks,  $T_p$  be the primary task,  $\{T_1, \dots, T_n\}$  be the task set we consider, and  $Unif\{T_1, \dots, T_n\}$  be a discrete uniform distribution over actions/tasks. To draw the  $k$ -th sample in a batch at time step  $t$ , we sample a  $p \in \mathbb{R}$  from a continuous uniform distribution  $Unif(0, 1)$ , and then sample an action (choice of task) as follows.

$$a_t^{(k)} \sim \begin{cases} Unif\{T_1, \dots, T_n\}, & \text{if } p < \epsilon \\ \{T_p\}, & \text{if } \epsilon \leq p < \gamma \\ \pi(\cdot | s_t^{(k)}), & \text{otherwise} \end{cases}$$

Then we sample from the data associated with the task.

Table 6.2: Performance improvements on low-resources GLUE tasks, RTE, and MPRC, when using high-resource GLUE tasks. Our proposed methods, both the time-varying version and the time-invariant version, outperform various commonly used data re-weighting baselines.

Auxiliary	Tasks	Single Task	Multi-task uniform	Multi-task proportional	Multi-task time variant	Multi-task time invariant
QNLI	RTE	0.675	0.7369	0.6679	0.7112	0.7473
QNLI	MRPC	0.8284	0.8652	0.7279	0.8653	0.8627
MNLI	RTE	0.675	0.6787	0.7401	0.7148	0.8014
MNLI	MRPC	0.8284	0.8358	0.7696	0.8603	0.8603

## 6.4 Experiments

We use the public BERT-base<sup>1</sup> [DCLT18] checkpoints as our base model to run multi-task fine-tuning experiments. We consider two scenarios for knowledge transfer: cross-lingual transfer between languages of the same task and cross-task transfer between tasks of the same language.

For the cross-lingual transfer study, we use the Cross-lingual Natural Language Inference (XNLI) corpus [CLR<sup>+</sup>18]. The data set asks whether a premise sentence entails, contradicts, or is neutral toward a hypothesis sentence. Crowd-sourced English data are translated to ten other languages by professional translators and used for evaluation, while the English MultiNLI [WNB18] training data and its translation to other languages, *translate-train* [HRS<sup>+</sup>20], are used for training. For the cross-task transfer study, we cover multiple tasks from the GLUE benchmark [WSM<sup>+</sup>19].

For all of our experiments, we compare two of our proposed methods to three baselines: uniform sampling which uses a uniform weight to sample across all tasks or languages, proportional sampling and the target task baseline which only uses the labeled data from the target task for training.

---

<sup>1</sup> <https://github.com/google-research/bert>

### 6.4.1 Training Settings

For all of our experiments, the classification model is simply a linear projection on top of BERT’s classification ([CLS]) token. One meta-step contains 1000 steps. During fine-tuning, we use batch size 64 and ADAM optimizer [KB17a] with learning rate  $2e^{-5}$ . For each REINFORCE update, we use a learning rate of 0.001 and run stochastic gradient descent for 100 steps. We train all models on TPU v3 with 8 cores. All code is implemented in Tensorflow.

### 6.4.2 Experimental Results

*Multilingual BERT* Based on the performance of multilingual BERT on XNLI, four languages consisting of Urdu, Swahili, Thai, and Hindi which are either low-resource or had low performance were picked. Single-task baseline denotes the classification performance of multilingual BERT on these languages with their respective translate-train data.

Table 6.1 shows the accuracy comparisons of the proposed methods on the languages picked from the XNLI dataset. Note that uniform sampling is the same as proportional sampling because the data set size is the same for all languages. From this table, we see that multi-tasking with naive uniform sampling performs worse than even target language baselines. Our methods, on the other hand, can utilize the auxiliary tasks to achieve quality improvements over single-task baselines. Interestingly, time-invariant policies in general have a similar quality to time-varying policies. This may indicate time-varying schedule used by previous work [GPB19] may not be necessary.

The sampling schedule is shown in Figure 6.1 with meta-steps on the X-axis and weights of languages on the Y-axis. We see the model outperforms the baselines by properly combining English supervised data or the target language’s translate-train data. We also observe

that a good sampling schedule can vary from task to task. It validates our motivation to learn the sampling schedule instead of using a rule-based one.

*English BERT* For GLUE, we select two high-resource tasks MNLI and QNLI with more than 100k labeled data each as auxiliary tasks. We target cross-task transfer to low-resource tasks: MPRC and RTE.

The results are shown in Table 6.2. Unlike the results on XNLI, uniform sampling, which naively up-samples the low-resource tasks, generally gives better accuracy than single-task baseline as shown in previous work [LHCG19]. We show that the sampling policy learned by our algorithm consistently outperforms all rule-based alternatives.

## 6.5 Conclusions

We propose a simple and effective RL-based algorithm to learn a static sampling schedule for multi-task learning, with the goal to improve the performance of one target task. We apply the algorithms to model fine-tuning and show that the proposed approach outperforms popular rule-based baselines.

## Masking-based OOD Detection

While pre-trained large-scale deep models have garnered attention as an important topic for many downstream natural language processing (NLP) tasks, such models often make unreliable predictions on out-of-distribution (OOD) inputs. As such, OOD detection is a key component of a reliable machine learning model for any industry-scale application. Common approaches often assume access to additional OOD samples during the training stage, however, outlier distribution is often unknown in advance. Instead, we propose a post hoc framework called *POORE - POsthoc pseudo Ood REgularization*, that generates pseudo-OOD samples using in-distribution (IND) data. The model is fine-tuned by introducing a new regularization loss that separates the embeddings of IND and OOD data, which leads to significant gains on the OOD prediction task during testing. We extensively evaluate our framework on three real-world dialogue systems, achieving new state-of-the-art in OOD detection.

## 7.1 Introduction

Although OOD detection has attracted a great deal of interest from the research community [GSS14, HG17, LLLS18], these approaches are not specifically designed to leverage the structure of textual inputs. Consequently, commonly used OOD approaches often have limited success in real-world NLP applications. Most prior OOD methods for NLP systems [LMP<sup>+</sup>19, CY21, KJL20] typically assume additional OOD data for outlier exposure [HMD18]. However, such methods risk overfitting to the chosen OOD set, while making the assumption that a relevant OOD set is available during the training stage. Other methods [GAEG20, LLS<sup>+</sup>21, KJL20] assume training a calibration model, in addition to the classifier, for detecting OOD inputs. These methods are computationally expensive as they often require re-training the model on the downstream task.

Motivated by the above limitations, we propose a framework called POORE that generates pseudo-OOD data using the trained classifier and the IND samples.

In summary, we make the following contributions:

1. We propose a Mahalanobis-based context masking scheme for generating pseudo-OOD samples that can be used during the fine-tuning.
2. We introduce a new POR loss that maximizes the distance between IND and generated pseudo-OOD samples to improve the OOD detection.
3. Though extensive experiments on the three benchmarks, we show that our approach performs significantly better than existing baselines.

## 7.2 Related Works

**OOD Detection.** It is a binary classification problem that seeks to identify unfamiliar inputs during inference from in-distribution (IND) data observed during training. Standard OOD methods can be divided into two categories. The first category [LLLS18, PLB<sup>+</sup>21, NMTL19, RLF<sup>+</sup>19] corresponds to approximating a density  $p_{IND}(x)$ , where density is used as a confidence estimate for binary classification. The second category of approaches [HG16, HG17, LMS<sup>+</sup>17, GG16] uses the predictive probability to estimate the confidence scores. In our experiments, we compare approaches from both categories.

**OOD Detection in NLP.** There have been several methods developed for OOD detection in NLP. [LLS<sup>+</sup>21] proposed using  $k$  sub-models, where each model is trained with different masked inputs. [KJL20] uses an external OOD set to train an additional calibration model for OOD detection. Most related to our proposed framework is MASKER [MML<sup>+</sup>21] that leverages IND data to generate pseudo-OOD samples, and uses self-supervision loss inspired from [DCLT18] and predictive entropy regularization for pseudo-OOD inputs. We also use BERT self-supervision-inspired keyword masking, however, we propose a novel keyword selection criterion. Moreover, we also introduce a novel model of regularization loss that directly increases the distance of IND and pseudo-OOD samples.

## 7.3 Preliminaries and Notations

We consider a deep learning model  $g \circ f(x)$  composed of an encoder  $f : \mathcal{X} \rightarrow \mathcal{F}$  and a classifier  $g$  that maps  $f(x)$  to the output space, where  $x \in \mathcal{X}$  corresponds to natural sentences composed of a sequence of tokens  $v_i \in \mathcal{V}$ , *i.e.*  $x = [v_1, \dots, v_T]$ ,  $T$  is the length of the sequence, and  $\mathcal{V}$  is the token vocabulary. For a downstream classification task, the

class prediction is defined as  $p(y|x) = \text{softmax}(g(f(x)))$ .

**Architecture.** In this work, we construct  $f$  using the bi-directional Transformer architecture [VSP<sup>+</sup>17]. Specifically, we use the encoder architecture proposed in [DCLT18] such that  $f(x)$  is the final hidden representation of the CLS token. We use a two-layer multi-layer perceptron (MLP) as the classifier  $g$ .

**Mahalanobis OOD Scoring.** OOD methods typically learn a confidence estimator that outputs a score  $s(x) \in \mathbb{R}$  such that  $s(x^{ind}) > s(x^{ood})$ , where  $x^{ind}$  and  $x^{ood}$  are sampled from IND distribution ( $\mathcal{D}_{IND}$ ) and OOD distribution ( $\mathcal{D}_{OOD}$ ) respectively. [LLLS18] proposed using Mahalanobis distance estimator for OOD detection that uses pre-trained features of the softmax neural classifier. Namely, given the feature of a test sample  $\phi(x)$ , the Mahalanobis score  $s_M(x)$  is computed as follows

$$d(x, c) = (\phi(x) - \hat{\mu}_c)^T \hat{\Sigma}^{-1} (\phi(x) - \hat{\mu}_c) \quad (7.1)$$

$$s_M(x) = - \min_c d(x, c) \quad (7.2)$$

where  $\phi$  is an intermediate layer of the neural classifier and  $c$  denotes the class. The parameters of the estimator  $\{\hat{\mu}_c, \hat{\Sigma}\}$  denote the class-conditional mean and the tied covariance of the IND features.

## 7.4 Post hoc Pseudo-OOD Regularization

In this section, we describe our framework called POsthoc pseudo Ood REgularization (POORE), which uses pseudo-OOD samples for fine-tuning a pre-trained classifier. We first describe our masking-based approach to generate pseudo-OOD samples from the IND samples available during training. These generated pseudo-OOD samples are used to regularize the encoder during post-hoc training of a pre-trained classifier, which leads to

improved robustness of the model towards OOD samples.

#### 7.4.1 Masking for Pseudo-OOD Generation

We perform context masking of IND samples for generating pseudo OOD samples. To generate context-masked pseudo OOD samples, we first identify a set of tokens  $v \in \mathcal{K} \subset \mathcal{V}$  that have high attention scores and consequently, a higher influence in model predictions. Given the set of keywords, we perform random masking of non-keywords in a given IND sample  $x$  to generate a pseudo OOD sample  $\tilde{x}$ .

**Keyword Selection.** We follow the attention-based keyword identification method proposed in [MML<sup>+</sup>21]. The token importance is measured using average model attention values computed in the final layer of the pre-trained transformer encoder. While this approach generates context-deprived inputs, the identified tokens are uniformly selected from all the IND samples in the training data. Instead, we propose a novel weighting criterion for keyword selection, such that higher weight is given to the tokens belonging to the training inputs that have a higher distance from the overall IND distribution determined by  $\{\hat{\mu}_c, \Sigma\}$ . This encourages the selection of keywords that belong to IND inputs that are far from the estimated IND distribution. Specifically, we propose the importance score criterion as follows:

$$I_M(v) = \frac{1}{n_v} \sum_{x \in \mathcal{D}_{IND}} \left( \sum_{i=1}^{i=T} v_{i=v} \cdot a_i \right) \cdot \hat{s}_M(x), \quad (7.3)$$

$$\text{where } \hat{s}_M(x) = \frac{s_{max} - s_M(x)}{s_{max} - s_{min}}, \quad (7.4)$$

$$s_{min} = \min_{x \in \mathcal{D}_{IND}} s_M(x'), \quad (7.5)$$

$$s_{max} = \max_{x \in \mathcal{D}_{IND}} s_M(x'). \quad (7.6)$$

where  $a_i \in [0, 1]$  is the attention value for  $i^{th}$  token in the last self-attention layer in  $f$ . The keyword set  $\mathcal{K}$  is formed by selecting top  $M$  tokens based on the token importance score  $m(x)$ . Note that in (7.3), the tokens in the IND samples having a lower Mahalanobis score (*i.e.* a higher Mahalanobis distance) will be more likely to be selected as a keyword.

**Context Masking.** Given the set of keywords  $\mathcal{K}$ , pseudo OOD samples can be generated by randomly masking the context. The context in an input  $x$  refers to the non-keyword tokens  $v \notin \mathcal{K}$ . We randomly mask the context tokens to generate a pseudo-OOD input  $\tilde{x}$ , which we use for regularization to improve the model reliability towards OOD inputs. More specifically, we create  $\tilde{x} = [\tilde{v}_1, \dots, \tilde{v}_T]$  as follows

$$u \sim \text{Uniform}(0, 1) \quad (7.7)$$

$$\tilde{v}_i = \begin{cases} \text{MASK} & \text{if } p_{mask} \leq u \\ v_i & \text{otherwise} \end{cases} \quad (7.8)$$

where MASK is the masking token,  $p_{mask}$  is the masking probability and  $v_i$  is the  $i^{th}$  token in the corresponding IND sample  $x$ .

### 7.4.2 Pseudo-OOD Regularization

To increase the Mahalanobis distance of OOD inputs relative to their IND counterparts, we propose Pseudo Ood Regularization (POR) loss. The POR regularization maximizes the distance between the IND sample and its corresponding pseudo-OOD sample. The POR loss  $L_{POR}$  is defined as

$$\mathcal{L}_{POR} = -_{x \sim \mathcal{D}_{IND}} \|f(x) - f(\tilde{x})\|_2 \quad (7.9)$$

where  $\tilde{x}$  the context-masked version of  $x$ .

**Post hoc Training.** The total loss used during post hoc fine-tuning is defined as

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_{SKL} \cdot \mathcal{L}_{SKL} + \lambda_{POR} \cdot \mathcal{L}_{POR} \quad (7.10)$$

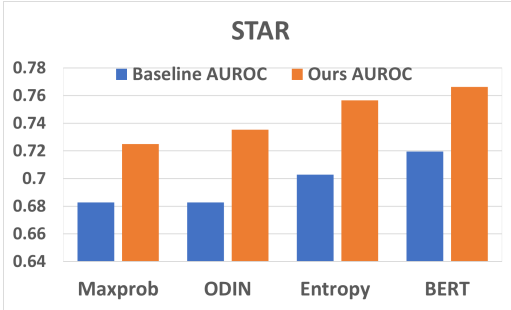
where  $\mathcal{L}_{CE}$  is the standard cross-entropy (CE) loss, and  $\mathcal{L}_{SKL}$  is the self-supervised keyword loss (SKL) proposed in [DCLT18]. The SKL loss has been found to improve the generalization of the model by avoiding overfitting of the model to certain tokens in the training data [MML<sup>+</sup>21]. The post hoc training process finetunes the model using (7.10). Note that the post hoc fine-tuning is carried on a model previously trained on the downstream task using the standard  $\mathcal{L}_{CE}$  loss.

## 7.5 Experiments

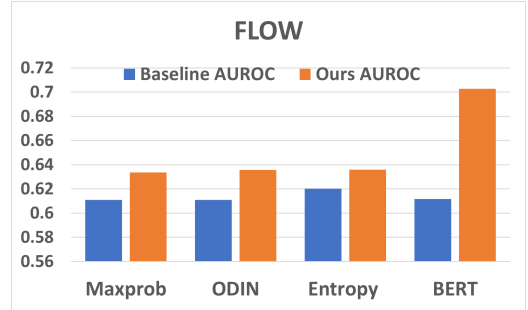
We demonstrate the effectiveness of our proposed approach in this section. For reproducibility of the experiments, we include the codebase in the supplementary.

Table 7.1: AUROC and FPR@90 of Baseline and POORE on the three target benchmarks. MASKER uses Maxprob for inference, MASKER-Maha and POORE use Mahalanobis for OOD detection during inference.

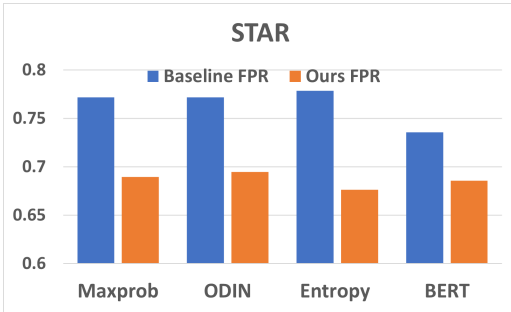
Methods	STAR		FLOW		ROSTD	
	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$
Maxprob [HG17]	68.27	77.18	61.10	84.23	91.49	54.30
Dropout [GG16]	52.77	100.0	51.86	100.0	55.25	100.0
Entropy [LG94]	70.29	77.84	62.02	79.45	91.86	53.83
Gradient Embed	67.61	80.80	71.21	70.25	98.53	2.58
BERT Embed [PLB <sup>+</sup> 21]	71.96	73.56	61.16	87.26	98.88	2.27
Mahalanobis [LLLS18]	76.89	65.25	73.13	<b>63.84</b>	99.45	1.00
MASKER [MML <sup>+</sup> 21]	71.54	72.82	68.16	67.52	86.95	54.26
MASKER-Maha (Enhanced)	79.38	59.97	72.99	65.23	99.41	1.15
POORE (Ours)	<b>81.11</b>	<b>48.30</b>	<b>74.08</b>	69.26	<b>99.51</b>	<b>0.97</b>



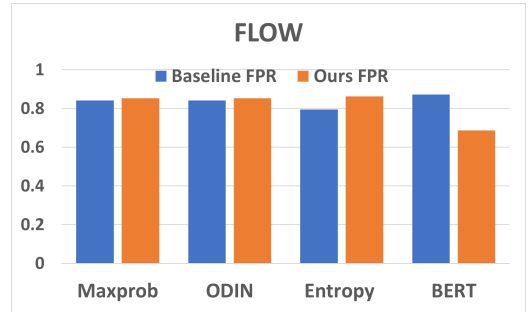
(a) AUROC for STAR



(b) AUROC for FLOW



(c) FPR@90 for STAR



(d) FPR@90 for FLOW

FIGURE 7.1: AUROC ( $\uparrow$ ) and FPR@90 ( $\downarrow$ ) using Maxprob, ODIN, Entropy, and BERT. The average improvements across estimators on AUROC are 5% on STAR (7.1a) and 4% on FLOW (7.1b). The FPR@90 improvements on an average are 8% on STAR (7.1c) and 2% on FLOW (7.1d).

### 7.5.1 Datasets

We use three task-oriented dialogue datasets for OOD detection. Namely, Schema-guided Dialog Dataset for Transfer Learning (STAR) [MMK20], SM Calendar flow (FLOW) [ABB<sup>+</sup>20], and Real Out-of-Domain Sentence From Task-oriented Dialog (ROSTD) [GAEG20]. We follow the data splits and pre-processing steps as described in [CY21]. A detailed description of these tasks is provided in Section 7.6.

### 7.5.2 Experimental Setup

Our approach is demonstrated on the BERT pre-trained model [DCLT18] with around 110M parameters trained on a single Titan-X GPU. We optimize  $\lambda_{POR}$  for each task through grid search from  $1e-5$  to  $1e2$  by a factor of 10, and use  $1, 1e-2, 1e-5$  for STAR, FLOW, and ROSTD respectively.

This paper compares our approach (POORE) with existing baseline OOD detection methods including Maxprob, Entropy, Mahalanobis, BERT Embed, Gradient Embed, and Dropout. We also consider MASKER [MML<sup>+</sup>21] as a baseline. A detailed analysis of the differences in performance between all the inference methods and our approach appears in Section 7.5.3. The baseline methods are trained for 25 epochs using the AdamW optimizer with a learning rate of  $1e-5, 3e-5, 1e-5$  for STAR, FLOW, and ROSTD respectively. We use minimal post hoc fine-tuning of only 1 additional epoch for MASKER and POORE. We use AUROC and FPR@90 to evaluate OOD detection performance. For more details on these metrics, refer to Section 7.7.

### 7.5.3 Results

Table 7.1 shows the performance gains from our approach relative to all the baseline methods on three target tasks namely STAR, FLOW, and ROSTD. POORE outperforms existing evaluation baselines by significant margins. Specifically on the STAR dataset, relative to Bert Embed and Mahalanobis baselines, we observe 9% and 4% respective absolute improvement in AUROC, while observing 26% and 17% respective absolute reduction in FPR@90. Similarly on FLOW, the AUROC gains were 13%, and 1% relative to BERT Embed and Mahalanobis, while doing worse only on the FPR@90 metric compared to the Mahalanobis baseline. We noted similar consistent gains on the ROSTD.

We also evaluate our framework POORE by pairing it with other confidence estimators like Maxprob, ODIN [LLS17], Entropy, and BERT Embed. Figure 7.1 compares a model trained using POORE with a standard model while using various confidence estimators during inference. As shown in figure 7.1, we observe significant gains with our framework over the baseline model for all the confidence estimators. Specifically, the AUROC on FLOW using Bert Embed with POORE achieved an improvement of 9%. We also pair the above estimators with the MASKER baseline and evaluate these combinations in the ablation shown in Section 7.8. In Section 7.9, we show an ablation comparing our novel keyword selection criterion with the keyword selection criterion used in the MASKER baseline.

## 7.6 Tasks

**STAR.** This is a dialog dataset with 6651 dialogues spanning multiple domains and intents [MMK20]. Responses to dialogs that were marked either “ambiguous” or “out-of-scope”

are used as OOD examples. The dataset has 29,104 examples with 104 intent labels.

**FLOW.** The FLOW dataset is a semantic parsing dataset with annotations for each turn of a dialog [ABB<sup>+</sup>20]. In FLOW, the OOD samples are from discussions where the user stays far away from the central topic. The dataset has 71,551 examples spanning 44 intents.

**ROSTD.** [GAEG20] designed ROSTD, a dataset proposed for OOD detection. They use external source for OOD samples, while the internal data represents IND. ROSTD contains 47,913 examples with 13 classes.

## 7.7 OOD Evaluation Metrics

OOD detection is evaluated using the Area Under the Receiver-Operating Curve (AUROC) metric for the binary classification task based on the estimated confidence score. An OOD method that perfectly separates  $s(x^{ind})$  from  $s(x^{ood})$  achieves an AUROC score of 100%. Another common metric used to evaluate OOD detection is false positive rate (FPR) at a fixed recall.

## 7.8 Adaptation of MASKER Baseline

In the table 7.2, improvised Masker baseline results can be seen, which include the results on a number of evaluation metrics. While the performance using an improvised baseline is better than the GOLD baseline, our approach beats this model considerably.

Table 7.2: Masker baseline and Adapted approaches.

Methods	STAR		FLOW		ROSTD	
	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$
Maxprob	71.54	7282	68.16	67.52	86.95	54.26
ODIN	72.45	71.86	68.57	66.81	86.86	54.25
BERT	75.93	62.89	69.79	70.07	99.16	1.73
Mahalanobis	79.38	59.97	72.99	65.23	99.41	1.15
POORE (Ours)	81.11	48.30	74.08	69.26	99.51	0.97

Table 7.3: Ablation for keywords

Methods	STAR		FLOW		ROSTD	
	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$	AUROC $\uparrow$	FPR@90 $\downarrow$
POORE with baseline keywords	80.69	58.71	73.41	68.08	99.52	1.06
POORE (Ours)	81.11	48.30	74.08	69.26	99.51	0.97

## 7.9 Ablation for choosing keywords

Table 7.3 compares the OOD detection performance of our proposed keyword selection approach described in Section 7.4.1 with the keyword selection criterion in the baseline MASKER in our proposed POORE framework.

## 7.10 Conclusions

In this paper, we propose a novel framework, which we call POORE, for improving the robustness of the model toward OOD data. Using a combination of Mahalanobis distance and POR regularization that maximizes the distance between IND and OOD representations, we demonstrated significant performance gains in a number of target benchmark tasks. Further work could tap into the potential of using external OOD data to achieve even more gains over other baselines that use outlier exposure.

## Conclusions

There has been significant progress in recent years in the study of attention-based models in a wide range of tasks, such as summarization, paraphrasing, and translation. Structure and feedback have significantly improved these models' ability to adapt under low-resource, multi-task, and other conditions. My dissertation outlines the research contributions I have made in this area, as well as significant performance improvements obtained using the proposed methods.

In Chapter 2, I presented the impact syntax of a language can have on translation using attention-based models. Furthermore, this was extended to BERT and evaluated on downstream tasks such as language inference and paraphrasing. For ablation, the experiments were conducted on both low-resource and high-resource languages.

In Chapter 3, I discussed how word order and syntax can play a significant role in machine translation across a variety of languages. Analysis of the effectiveness of the method was carried out by comparing recurrent networks with attention-based models.

In Chapters 4 and 5, I discussed my efforts to incorporate numeracy to word embed-

dings, which is overlooked by NLP models. There have been significant gains in numeracy tasks, including question answering and classification, when numbers are treated differently from word tokens. The effects of numeracy on named entity recognition are discussed specifically in Chapter 5.

Chapter 6 discusses my contribution to improving target task performance using feedback signals from a supporting model. Using this model, which weights individual tasks, target language performance is significantly enhanced.

Chapter 7 presents a method of masking tokens along with a novel loss function to maximize the distance between in-domain and pseudo-OOD samples. This resulted in a robust model which performed significantly better on OOD detection on three benchmarks.

In the future, the impact of the structure and feedback could be extended to other models like conversational AI. Additionally, it will be interesting to analyze whether using structure in data-starved settings like federated learning and continual learning is beneficial.

# Bibliography

- [AAH<sup>+</sup>09] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.
- [ABB<sup>+</sup>20] Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, 2020.
- [AG07] Galen Andrew and Jianfeng Gao. Scalable training of  $L_1$ -regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40, 2007.
- [AHLP19] Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. Giving bert a calculator: Finding operations and arguments with reading comprehension. *arXiv preprint arXiv:1909.00109*, 2019.
- [AJF19] Roei Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. *arXiv preprint arXiv:1903.00089*, 2019.
- [Ame83] American Psychological Association. *Publications Manual*. American Psychological Association, Washington, DC, 1983.
- [AU72] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.
- [AZ05] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, December 2005.
- [AZ12] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.
- [BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

- [BCDG09] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.
- [BDS03] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003.
- [BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [Bha07] Shishir Bhattacharja. Word formation in bengali: A whole word morphological description and its theoretical implications. 2007.
- [BJ<sup>+</sup>11] Benjamin Borschinger, Mark Johnson, et al. A particle filter algorithm for Bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 10–18, Canberra, Australia, 2011.
- [Blu60] LE Blumenson. A derivation of n-dimensional spherical coordinates. *The American Mathematical Monthly*, 67(1):63–66, 1960.
- [BS17] Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*, 2017.
- [BTA<sup>+</sup>17] Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima’an. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*, 2017.
- [BZVL17] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*, 2017.
- [Car97] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [CBS<sup>+</sup>15] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.
- [CDA<sup>+</sup>17] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.
- [CGF12] Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268, 2012.
- [CH19] Anna Currey and Kenneth Heafield. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth*

*Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, 2019.

- [CHCC17] Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. Improved neural machine translation with a syntax-aware encoder and decoder. *arXiv preprint arXiv:1707.05436*, 2017.
- [CHTC19] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. Numeracy-600k: Learning numeracy for detecting exaggerated information in market comments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6307–6313, 2019.
- [CKG<sup>+</sup>19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.
- [CKLM19] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert’s attention. *arXiv preprint arXiv:1906.04341*, 2019.
- [CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.
- [CLR<sup>+</sup>18] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*, 2018.
- [CM05] Barry R Chiswick and Paul W Miller. Linguistic distance: A quantitative measure of the distance between english and other languages. *Journal of Multilingual and Multicultural Development*, 26(1):1–11, 2005.
- [CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.
- [Cur05] George Oliver Curme. *A grammar of the German language*. Macmillan, 1905.
- [CVMG<sup>+</sup>14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.
- [CY21] Derek Chen and Zhou Yu. Gold: improving out-of-scope detection in dialogues using data augmentation. *arXiv preprint arXiv:2109.03079*, 2021.

- [CZZZ18] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs, 2018.
- [DB05] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [DL11] Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*, pages 85–91, 2011.
- [DMRM08] Marie-Catherine De Marneffe, Anna N Rafferty, and Christopher D Manning. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047, 2008.
- [DWD<sup>+</sup>19] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.
- [EMH18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.
- [fCM83] Association for Computing Machinery. *Computing Reviews*, 24(11):503–512, 1983.
- [Fir57] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.
- [For13] Montserrat Forcadell. Subject informational status and word order: Catalan as an svo language. *Journal of pragmatics*, 53:39–63, 2013.
- [GAEG20] Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7764–7771, 2020.
- [GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [GCO<sup>+</sup>19] Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc’Aurelio Ranzato. The flores evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. *arXiv preprint arXiv:1902.01382*, 2019.

- [GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [Gle98] Christopher Gledhill. *The Grammar of Esperanto*. 1998.
- [GMS<sup>+</sup>20] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don’t stop pretraining: Adapt language models to domains and tasks, 2020.
- [GPB19] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Autosem: Automatic task selection and mixing in multi-task learning. *arXiv preprint arXiv:1904.04153*, 2019.
- [GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.
- [GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [Gus97] Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK, 1997.
- [HAC<sup>+</sup>18] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.
- [Har54] ZS Harris. Distributional structure. *word*, 10 (2-3): 146–162. reprinted in *fodor, j. a and katz, jj (eds.), readings in the philosophy of language*, 1954.
- [HDU11] José Ignacio Hualde and Jon Ortiz De Urbina. *A grammar of Basque*, volume 26. Walter de Gruyter, 2011.
- [Hew95] Brian George Hewitt. *Georgian: A structural reference grammar*, volume 2. John Benjamins Publishing, 1995.
- [HFH17] Harald Hammarström, Robert Forkel, and Martin Haspelmath. Glottolog 3.0. *Max Planck Institute for the Science of Human History*, 2017.
- [HG16] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.
- [HG17] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

- [HLGC20] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.
- [HLW<sup>+</sup>20] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedziec, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.
- [HM16] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.
- [HMD18] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.
- [HML<sup>+</sup>16] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. *ArXiv*, abs/1603.06318, 2016.
- [HP<sup>+</sup>02] Rodney Huddleston, Geoffrey K Pullum, et al. The cambridge grammar of english. *Language*. Cambridge: Cambridge University Press, 1:23, 2002.
- [HRS<sup>+</sup>20] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization, 2020.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [HWH<sup>+</sup>17] Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. Towards neural phrase-based machine translation. *arXiv preprint arXiv:1706.05565*, 2017.
- [HXTS16] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.
- [IC17] Jinbae Im and Sungzoon Cho. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*, 2017.
- [Jen07] Marta Jenkala. *Read Ukrainian! A reading course for beginners*, 2007.
- [JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
- [JMC<sup>+</sup>16] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

- [JNG<sup>+</sup>19] Chengyue Jiang, Zhonglin Nian, Kaihao Guo, Shanbo Chu, Yinggong Zhao, Libin Shen, Haofen Wang, and Kewei Tu. Learning numeral embeddings. *arXiv preprint arXiv:2001.00003*, 2019.
- [JSL<sup>+</sup>17a] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google’s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.
- [JSL<sup>+</sup>17b] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s multilingual neural machine translation system: Enabling zero-shot translation, 2017.
- [JSS<sup>+</sup>19] Ganesh Jawahar, Benoît Sagot, Djamé Seddah, Samuel Unicomb, Gerardo Iñiguez, Márton Karsai, Yannick Léo, Márton Karsai, Carlos Sarraute, Éric Fleury, et al. What does bert learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy*, 2019.
- [JZ07] D Jan and Ludger Zeevaert. *Receptive multilingualism: Linguistic analyses, language policies and didactic concepts*, volume 6. John Benjamins Publishing, 2007.
- [KB13] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.
- [KB17a] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [KB17b] Tom Kocmi and Ondřej Bojar. An exploration of word embedding initialization in deep-learning tasks. *arXiv preprint arXiv:1711.09160*, 2017.
- [KDH<sup>+</sup>18] Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, 2018.
- [KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.
- [KGS<sup>+</sup>17] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

- [KJL20] Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. *arXiv preprint arXiv:2006.09462*, 2020.
- [KKD<sup>+</sup>17a] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.
- [KKD<sup>+</sup>17b] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.
- [Koe05] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer, 2005.
- [Kou08] Omkar N Koul. *Modern Hindi Grammar*. Dunwoody Press Springfield, USA, 2008.
- [KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [KZS<sup>+</sup>15] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.
- [LBS<sup>+</sup>16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
- [LCF18] Surafel M Lakew, Mauro Cettolo, and Marcello Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. *arXiv preprint arXiv:1806.06957*, 2018.
- [LDG16] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016.
- [LG94] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR’94*, pages 3–12. Springer, 1994.
- [LHCG19] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding, 2019.
- [LLS18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.
- [LLS<sup>+</sup>15] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

- [LLS17] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.
- [LLS<sup>+</sup>21] Xiaoya Li, Jiwei Li, Xiaofei Sun, Chun Fan, Tianwei Zhang, Fei Wu, Yuxian Meng, and Jun Zhang.  $k$  folden:  $k$ -fold ensemble for out-of-distribution detection. *arXiv preprint arXiv:2108.12731*, 2021.
- [LM15] Minh-Thang Luong and Christopher D Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, 2015.
- [LMP<sup>+</sup>19] Stefan Larson, Anish Mahendran, Joseph Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. *ArXiv*, abs/1909.02027, 2019.
- [LMS<sup>+</sup>17] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.
- [LOG<sup>+</sup>19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- [LSHL20] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [LSR<sup>+</sup>18] Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li, and Qi Su. Learning when to concentrate or divert attention: Self-adaptive attention temperature for neural machine translation. *arXiv preprint arXiv:1808.07374*, 2018.
- [LT89] Charles N Li and Sandra A Thompson. *Mandarin Chinese: A functional reference grammar*, volume 3. Univ of California Press, 1989.
- [LT16] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. 2016.
- [LTY<sup>+</sup>18] Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. Multi-head attention with disagreement regularization. *arXiv preprint arXiv:1810.10183*, 2018.

- [LXT<sup>+</sup>17] Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. Modeling source syntax for neural machine translation. *arXiv preprint arXiv:1705.01020*, 2017.
- [Mat97] Terje Mathiassen. *A short grammar of Latvian*. Slavica Pub, 1997.
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [MKXS18] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.
- [ML09] Jeff Mitchell and Mirella Lapata. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. Association for Computational Linguistics, 2009.
- [MM07] Bill MacCartney and Christopher D Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200. Association for Computational Linguistics, 2007.
- [MMK20] Johannes EM Mosig, Shikib Mehri, and Thomas Kober. Star: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*, 2020.
- [MML<sup>+</sup>21] Seung Jun Moon, Sangwoo Mo, Kimin Lee, Jaeho Lee, and Jinwoo Shin. Masker: Masked keyword regularization for reliable text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13578–13586, 2021.
- [MSC<sup>+</sup>13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [MSGH16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.
- [MZC<sup>+</sup>19] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H. Chi. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *AAAI*, pages 216–223, 2019.
- [NC17] Jan Niehues and Eunah Cho. Exploiting linguistic resources for neural machine translation using multi-task learning. *arXiv preprint arXiv:1708.00993*, 2017.

- [NM19] U. Naseem and K. Musial. Dice: Deep intelligent contextual embedding for twitter sentiment analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 953–958, 2019.
- [NMTL19] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *ArXiv*, abs/1906.02994, 2019.
- [NRRH19] Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380, 2019.
- [PB17] Ramakanth Pasunuru and Mohit Bansal. Multi-task video captioning with video and entailment generation. *arXiv preprint arXiv:1704.07489*, 2017.
- [PBS15] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.
- [PLB<sup>+</sup>21] Alexander Podolskiy, Dmitry Lipin, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. Revisiting mahalanobis distance for transformer-based out-of-domain detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13675–13682, 2021.
- [PNI<sup>+</sup>18] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.
- [PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- [PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [PSN<sup>+</sup>19] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [RBAS19] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

- [Res12] Seymour Resnick. *Essential French Grammar*. Courier Corporation, 2012.
- [Rey80] Christopher Hanby Baillie Reynolds. *Sinhalese, an introductory course*. School of Oriental and African Studies, University of London, 1980.
- [RHW<sup>+</sup>88] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [RLF<sup>+</sup>19] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. *Likelihood Ratios for Out-of-Distribution Detection*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- [RSR<sup>+</sup>20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.
- [RT15] Mohammad Sadegh Rasooli and Joel R. Tetreault. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733, 2015. version 2.
- [RZLL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [SCS<sup>+</sup>19] Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, Asli Celikyilmaz, and Lawrence Carin. Learning compressed sentence representations for on-device text processing. *arXiv preprint arXiv:1906.08340*, 2019.
- [SDCW19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [SDM03] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.
- [SG16] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, 2016.
- [SH16] Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*, 2016.
- [SHB15] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

- [SK81] B Syamala Kumari. An intensive course in malayalam, 1981.
- [SMS<sup>+</sup>99] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer, 1999.
- [Soh01] Ho-Min Sohn. *The korean language*. Cambridge University Press, 2001.
- [SPW<sup>+</sup>06] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. *arXiv preprint cs/0609058*, 2006.
- [SPW<sup>+</sup>13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- [SR18] Georgios P Spithourakis and Sebastian Riedel. Numeracy for language models: Evaluating and improving their ability to predict numbers. *arXiv preprint arXiv:1805.08154*, 2018.
- [SS21] Vivek Subramanian and Dhanasekar Sundararaman. How do lexical semantics affect translation? an empirical study. *arXiv preprint arXiv:2201.00075*, 2021.
- [SSS<sup>+</sup>20] Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753, 2020.
- [SSW<sup>+</sup>19] Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*, 2019.
- [SSW<sup>+</sup>21] Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. Syntactic knowledge-infused transformer and bert models. In *CIKM Workshops*, 2021.
- [SSW<sup>+</sup>22] Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Liyan Xu, and Lawrence Carin. Number entity recognition. *arXiv preprint arXiv:2205.03559*, 2022.
- [STL<sup>+</sup>21] Dhanasekar Sundararaman, Henry Tsai, Kuang-Huei Lee, Iulia Turc, and Lawrence Carin. Learning task sampling policy for multitask learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4410–4415, 2021.

- [SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.
- [SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [SZH<sup>+</sup>18] Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. Deconvolutional latent-variable model for text sequence matching. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [TBC<sup>+</sup>17] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.
- [THR<sup>+</sup>18] Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8035–8044, 2018.
- [TP11] Michel Tokic and Günther Palm. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *Annual Conference on Artificial Intelligence*, pages 335–346. Springer, 2011.
- [Uni20] Universal Dependencies. *UD for Galician*, 2020.
- [VSP<sup>+</sup>17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [VWM<sup>+</sup>20] Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. Exploring and predicting transferability across nlp tasks, 2020.
- [WBGL15] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.
- [WGY18] Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision rnns for language recognition. *arXiv preprint arXiv:1805.04908*, 2018.
- [WLW<sup>+</sup>18] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

- [WNB17] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.
- [WNB18] Adina Williams, Nikita Nangia, and Samuel Bowman. 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)*, pages 1112–1122. Association for Computational Linguistics, 2018.
- [WPAN19] Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. Multilingual neural machine translation with soft decoupled encoding. In *International Conference on Learning Representations*, 2019.
- [WSB18] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.
- [WSC<sup>+</sup>16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
- [WSM<sup>+</sup>18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [WSM<sup>+</sup>19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.
- [WTN20] Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online, July 2020. Association for Computational Linguistics.
- [WWL<sup>+</sup>19a] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*, 2019.
- [WWL19b] Chengyi Wang, Shuangzhi Wu, and Shujie Liu. Source dependency-aware transformer with supervised self-attention. *arXiv preprint arXiv:1909.02273*, 2019.
- [XSdT17] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [YDL<sup>+</sup>18] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

- [YHSBK17] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org, 2017.
- [YTW<sup>+</sup>18] Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. Modeling localness for self-attention networks. *arXiv preprint arXiv:1810.10182*, 2018.
- [YZLL21] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.
- [ZCH20] Yinhe Zheng, Guanyi Chen, and Minlie Huang. Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 28:1198–1209, jan 2020.
- [ZDLS20] Ming Zhou, Nan Duan, Shujie Liu, and Heung-Yeung Shum. Progress in neural nlp: modeling, learning, and reasoning. *Engineering*, 6(3):275–290, 2020.
- [ZTS19] Biao Zhang, Ivan Titov, and Rico Sennrich. Improving deep transformer with depth-scaled initialization and merged attention. *arXiv preprint arXiv:1908.11365*, 2019.
- [ZVSL18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

# Biography

Dhanasekar Sundararaman is a Ph.D. candidate in the Electrical and Computer Engineering program at Duke University. His research interests include integrating structural information and developing feedback-based models for natural language processing. He has published several of his works in top-tier venues including ACL, EMNLP, and CIKM. During his Ph.D., he performed research in the industry at Google Research, Microsoft, and Amazon Science through internships. He also is serving as a reviewer for conferences including ACL, EMNLP, NAACL, COLING, and CIKM. Before coming to Duke, he received his Bachelor's from Anna University in India.