Generalized Model for Classifications and Q/A on Natural Languages

Md Momin Al Aziz
University of Manitoba
azizmma@cs.umanitoba.ca

Abstract

The recent advents in Recurrent Neural Networks on Natural Language Processing have facilitated the necessity towards a generalized pretrained model for arbitrary tasks on textual data. In this report, we take a closer look at some of the recent developments while proposing new methods to construct such a unified model. We propose two novel unsupervised tasks along with a graph-based word embedding. Our experimental results show the efficacy of these modalities while comparing them with the state-of-the-art solution for classification and question/answering tasks.

1. Introduction

Natural languages are a substantial component of our data which is often represented within the textual format. Computationally, Natural Language Processing (NLP) functions on these textual data by understanding the central idea of an arbitrary text document while retrieving valuable information. Regardless of the underlying task, there has always been a need for 'a shoe that fits all' where one pretrained machine learning model can answer arbitrary queries with some additional steps.

In this work, we intend to investigate two of these basic NLP queries (or tasks)— a) Classification and Question Answering (QA). Over the years, these tasks underwent a steady, foreseen improvement on the different dataset from multiple research areas. However, recent advancements in Deep Learning (DL) techniques has landscaped the performance of most pre-DL approaches.

In similitude to Computer Vision overtures towards a generalized pretrained model (*i.e.* ImageNet [17], ResNet [11] *etc.*) for arbitrary vision tasks, NLP has recently shifted towards this direction. This late migration has been attributed to the mainstream Long Short-Term Memory models (LSTMs) [12] or more recently poplar Transformer model [31]. Recent attempts such as BERT [7], ELMo [25] or OpenAI-GPT [26] utilize either LSTMs or transformers to attain the state-of-the-art (SOTA) in various NLP tasks.

However, to train a generalized LSTM or transformer model for arbitrary tasks, we require a sizeable number of labelled instances which happen to be in scarce. Thus, the aforementioned approaches utilized some un/semi-supervised tasks to train their astronomical models with million parameters. One key component in this task will be *Language Modelling* (LM). For example, BERT [7] utilized an LM model (detailed in Section 2.3) due to its simplicity and abundance of unlabeled text data. The proceeding, notable QA solutions such as OpenAI GPT [26], ELMo [25] also employed LM underneath. Practically, training any LM model on a specific dataset is somewhat seamless due to the abundance of unlabeled textual data. However, the quality of an LM model can only be judged with an underlying task (i.e. QA, translation, *etc.*). In this work, we look a little closer into the training of these LM tasks and add, modify them accordingly (more in Section 3.1).

The other key ingredient in any NLP task is *word embedding*. Broadly, word embedding denotes a vector for each word based on its syntax, semantics and context. There are several embedding available such as Word2Vec [10], ElMo [25], WordPiece [35] which are trained on different datasets and more importantly employed in many QA tasks. In this project, we propose another embedding scheme using Graph Auto-Encoders (GAE) [16].
In summary, we claim the following contributions:

- We demonstrate the importance of characteristics based Language Modeling (LM) where we mask specific words based on their named entity (Section 3.1.1).
- We add a new unsupervised task to our baseline model BERT [7] which utilized the order between two sentences and their membership on a certain document (Section 3.1.2).
- We successfully added a new word embedding scheme based on the dependency graphs where the intermittent word relations are preserved (Section 3.2).
- Our experimental analysis on a much smaller 1000 document dataset gain 81.17% accuracy on classification tasks compared to BERT's 85.81%; although falling short for the QA. The experimental results and discussions in Section 4 should lay the groundwork for the future directions for this work.

2. BERT Architecture

Since our model relies on Bert [7] (and its predecessors), its architecture needs to be discussed. Furthermore, this discussion will serve as preliminary background knowledge about the typical NLP pipeline and reflect the contrast (and similarity) between our proposed models and Bert as we describe our methods later in Section 3.

2.1. Language Input Representation

In NLP, textual inputs (*i.e.* documents, paragraphs, sentences *etc.*) are usually separated into distinct words. Thus, a text input can be represented as a collection of unique words and their convoluted relations. These words are traditionally represented by a static numeric embedding vector of a fixed size. Myriad NLP tasks have utilized two popular word embedding libraries— GloVe [24] and Word2Vec [21, 10] where vector representation of millions of unique words are available.

Tokenization. BERT's approach on treating words is somewhat contrary as it does not offer a fixed vectorization for each word. One of the primary reasons is words often contain suffixes (`ed, ing`) due to their variant usage and different parts of speech (POS). For example, `opening`, `opened`, `opener` all relate to one word `open`. Thus, BERT uses a *Word Piece Tokenizer* [35] which splits the words into two such as `open` and `##ing`, `##ed`, `##er` for the prior example. For the rest of the paper, we denote word piece tokens simply as tokens.

Embeddings. After tokenization, BERT assigns numeric embedding vectors E_T^i with size h for each i^{th} tokens. Another pretrained positional vector E_P^i is added with E_T. Here, E_P is a fixed vector of size $|512 \times E|$ as it only contains vectors for a fixed $i \in [0, 511]$ positions (further in Section 3.2). Lastly, another vectors for each sentence/segment E_S^i are added to $E_T^i + E_P^i + E_S^i$. In summary, for n sequence length the final embedding vector will be,

$$E = [E^1, E^2, \ldots E^n] \text{ where}$$
$$E^i = E_T^i + E_P^i + E_S^i$$

It is noteworthy that all three vectors E_T^i, E_S^i, E_P^i are of sized as $|VocabSize \times h|$, $|2 \times h|$ and $|512 \times h|$ respectively ($h = 768$ in Table 1). Hence, there are only a 2 segment embedding vectors denoting maximum 2 segments and fixed 512 positional embedding available.

Stop Words. The beginning of the input tokens are marked with `[CLS]` whereas all sentences have a stopping token `[SEP]` which separates the next sentence.

In summary, a text dataset is comprised of *documents* which comprises *sentences* and its words. The words are split into tokens which are the fundamental blocks and converted to fixed sized vectors for further processing.

Table 1: BERT model specifications

Parameter	BERT-Base	BERT-Large
Layer l	12	24
Hidden Size h	768	1024
Soft-Attention a	12	16
Total Parameters	110M	340M

2.2. Transformer Model

BERT architecture is based on soft-attention based transformer model [31]. Here, unit transformer encoders are layered on top of each other and connected bi-directionally. For example, in Figure 1, we have a n length sequence (n tokens) on which n transformers are placed at the first layer. Then, depending on the model size, m transformer layers are attached one after the other. Each transformer unit will connect with all the succeeding transformers of the next layer. The details of transformer encoders can be found in the original article [31] where it showed significant performance improvement over LSTMs.

Notably, the fundamental difference between OpenAI's GPT [26] and BERT is the bi-directional transformer connection whereas GPT only connects the rightwards units. This association is analogous to the (directional) semantic connection among words where bi-directional connections tend to be more expressive. Another significant approach ELMo [25] also utilized bi-directional connections employing LSTMs.

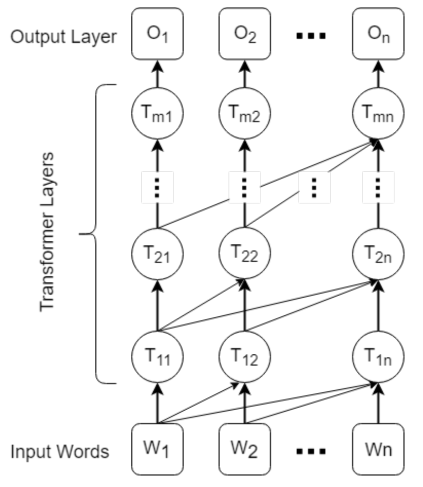
The input layers (Figure 1) take one vector as input for individual tokens. Hence, W_i in the figure are vector inputs whereas the outputs O_i's are single values. It is noteworthy that the output values O_i can be used for arbitrary tasks (*i.e.* translations, classifications, Question Answering, *etc.*).

In the original paper, BERT was analyzed on different hyperparameters which defined the size of the architecture. More specifically, the three major components of the model were the number of transformer layers l, hidden size h and soft-attention head size a. After a comprehensive analysis of these three parameters, the authors proposed two models presented in Table 1. Due to the size of these models and experimental breadth, we do not offer any structural modification to these in our approach.

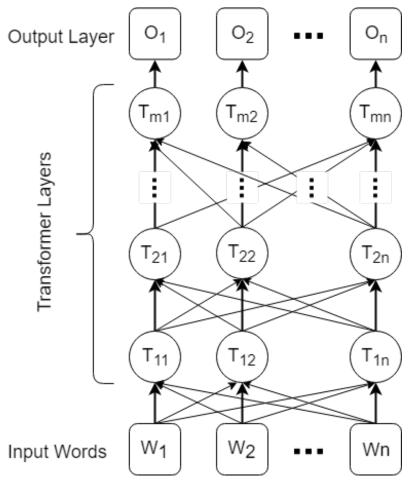
2.3. Pretraining Tasks

Initially, BERT is pretrained on a large dataset and later trained or fine-tuned for specific tasks such as Classifications, Translations *etc.* This pretraining is done unsupervised as it enables us to employ a significant amount of data to train the underlying model. This also empowers the model to learn a more generalized representation of the data and perform better for any specific tasks later on.

Thus, pretraining is unsupervised and training will be on



(a) OpenAI GPT model



(b) BERT (Bidirectional) model

Figure 1: Architectural difference between OpenAI GPT and BERT. BERT is fully connected (left to right) whereas OpenAI GPT only has rightwards connection. Both models have take in n word/tokens in m layers of soft-attention based Transformers.

supervised tasks hereafter. All BERT models comprising a million parameters were pretrained with only two tasks which are later extended in our approach as they are the central thesis in this paper:

Masked Language Modeling (MLM). In the Masked Language Modeling (MLM) task, random 15% tokens were picked and underwent the following pipeline:

- Replace the word with `[MASK]` 80% of the time
- Keep the original word on 10% training instances, and
- Replace with a random word for the remaining 10%.

It is important to understand that if we have a 80% probability of masking each word where only 15% words are masked in one sentence/segment, most of the earlier words in any sentence (subjects) will be picked. Hence, all the underlying segments in the dataset goes through the same masking process several times as the masking tokens are randomly chosen. This is an unsupervised task where the model needs to correctly guess the token on the `[MASK]`ed position. We elaborate further on this task in Section 3.1.

Next Sentence Prediction. BERT also targeted a popular NLP task which is named as *Entailment*. Textual Entailment [34] denotes whether the ordering or direction of two sentences is logical. However, BERT simplified this notion as the authors formulated the task as:

Problem Definition 1. *Given two sentences SEG_1 and SEG_2, classify whether SEG_2 is the next sentence right after SEG_1 (binary classification).*

Hence, for every sentence SEG_i in the training instances SEG_{i+1} is picked as its next sentence for 50% of the time. In the rest of the instances, a random sentence (not SEG_{i+1}) is placed as next. The classification label for both the instances is 1 and 0 respectively.

2.4. Pretraining and Finetuning Procedure

BERT was pretrained on the English Wikipedia and the Books Corpus [37] with *2,500 million and 800 million* words respectively (including other smaller corpus). The maximum sequence length considered consisted of 512 tokens with a batch size of 256. Hence, the model foresaw at least 256×512 tokens for each batch which is a staggering 128k tokens with an embedding size of at least $h = 768$ (Table 1).

This large scale pretraining on such astronomical dataset is only possible with 16 Tensor Processing Units (TPUs), proprietary to Google and readily available only on Google Cloud Platform. However, it is noteworthy that all the proposed method experimented in this paper were done in traditional GPU based system which significantly reduced the maximum allowed sequence length and batch size (details in Section 3.3).

The finetuning, on the contrary, was simpler operations where only the final output layer was modified (Figure 1b). For example, if we are building a binary classifier, we will add a fully connected layer on top of BERT's output layer with only two new output logits ($[768, 2]$). These outputs will denote the positive or negative instances for the corresponding binary classifier. This is an easy task compared to the earlier pretraining as it will only take 3/4 epochs where the original hyperparameters are kept the same [7].

3. Methods

In our proposed method we only keep the BERT Transformer architecture as described in Section 2.2 and perform additional pretraining tasks. More specifically, we do not remove any of the current architectural properties (size and units) but incorporate other methodologies elsewhere.

3.1. Pretraining Tasks

We refine the Masked LM (MLM) task and extend the Next Sentence Prediction task while we perform further pretraining on the original BERT model.

3.1.1 Entity based MLM (EMLM).

In BERT, the authors utilized a Masked LM (MLM) procedure which similar to a Cloze task where 15% words were taken out randomly (and put back in different instances) from their sentences as mentioned in Section 2.3. The model was later trained on a large corpus (*i.e.* Wikipedia) to predict the corresponding word. Thus, the resulting model will have an understanding of the text realizing the word distribution.

However, this precise word prediction is somewhat too coarse since the Clozes are picked randomly. For example, every word in a sentence does not have the same significance. Thus, in a random 15% masking, less significant words such as the determiners (`a`, `the`) or Prepositions (`in`, `of`, *etc*.) might be considered. Furthermore, the nouns or the subjects of the sentence carry more value towards understanding the semantics of the text. For example,

```
The University of Manitoba was established
in 1877 and is Western Canada's first
 university...
```

Here, we want our model to learn the name of the university (organization) and the year (1877) which is more significant in understanding the text. Thus, we integrated the Named Entities (NE) and the Parts of Speech while masking the original text before pretraining. Named Entities are a well-known and heavily utilized concept in Information Extraction as they denote real-world objects (*i.e.* Person, Organizations, Date, Locations *etc*.). NE Recognition (NER) is a well-sought problem in NLP where each word is classified into these entities. There are multiple state-of-the-art libraries available including BERT as it can be finetuned to perform NER.

However, we avoid the *chicken or the egg causality dilemma* (what comes first) as we use a NER library to get the entities of each word (specifically token) and modify the masking criteria as such:

- For any named entity and noun, we increase the masking probability to 50%
- Furthermore, the preceding and succeeding token of the NE/noun token also has its probability set at 50%
- For all other tokens, the probability is kept at 30% (reduced from 80%)
- If a token is not masked, 70% of the time the original token is placed in their rightful place
- The rest of 30% of the time it is replaced with a random token (previously it was 50%).

We utilize Spacy [1] as our NER library to remove the dependency and possible bias from BERT's NER. Also, we used the aforementioned masking technique to fine-tune or perform further pretraining on top of the BERT model increasing the maximum masking to 20% of the original sequence length to accommodate more masked token (prior 15%). Thus, the masked example will somewhat be:

```
The [MASK] of [MASK] was established
in [MASK] and is [MASK] Canada's first
 university...
```

3.1.2 Sentence Direction and Relation (SDR)

Along with the modified named entity based MLM and next sentence prediction task, we introduced another unsupervised task which reflects the logical entailment (`If A then B`) and the relations between them. This is posed as a 3-class classification task where for two sentences `SEG1` and `SEG2` in any document:

- **Class1**: `SEG2` can be placed after (entailed) `SEG1`
- **Class2**: `SEG2` should appear before `SEG1`, and
- **Class3**: `SEG1` and `SEG1` do not have any relation between them (from different document)

It is noteworthy that BERT considers two subsequent sentences on its next sentence prediction task whereas we pick two arbitrary sentences from one document. Thus, our proposed task can be exampled in Table 2.

3.2. Dependency Embedding in Inputs

Apart from the three embedding vectors from BERT, we add another embedding vector that represents the dependency among the tokens. Here, we utilize two concepts named, Dependency Grammar and Graph AutoEncoders (GAEs).

[1] https://spacy.io/

Table 2: Sentence Direction and Relation Task

`SEG1`	`SEG2`	Label
The University of Manitoba was established in 1877	The Manitoba Bisons represents the university in U Sports	0
The Manitoba Bisons represents the university in U Sports	The University of Manitoba was established in 1877	1
The University of Manitoba was established in 1877	Geoffrey Hinton, Yoshua Bengio, Yann LeCun won Turing awards in 2018	2

3.2.1 Dependency Grammar

Dependency Grammar is a part of formal grammar which illustrates the relationship between words in a sentence. This hierarchical dependency reflects the connection between the subject and the predicates based on their POS [2]. This allows us to formulate any sentence as a graph where each word/-token is connected with another which we utilize here.

Though this dependency parsing is a traditional NLP problem, it is best explained via an example in Figure 2. We perceive that `University` is connected to `established` as the subject. Furthermore, `Manitoba` is connected to `University` as they are clustered as a phrase. Lastly, the year `1877` is connected with `established` via a preposition.

There is also another property which makes this dependency grammar interesting as we can visualize the projection of any word. For example, in Figure 2, `Manitoba` is connected to `1877` via `University`, `established` and `in`. This relations can particularly useful in Question Answering tasks which we see later on.

3.2.2 Graph AutoEncoders (GAEs)

The information from these dependency graphs (specifically trees) are taken as inputs to Graph AutoEncoders (GAEs) [16]. We perform an unsupervised training of GAEs based on Graph Convolutional Network (GCNs) where all the edge connectivity were not present while training. For example, the word `University` (Figure 2) is connected with `of` whereas `of` is connected with `Manitoba`. Hence, there will be an edge from `University`$\rightarrow$ `of` and `of` $\rightarrow$ `Manitoba`.

The model was trained to detect these edge connections between words where words were represented as graph vertices V and their relations as edges E. The prepositions were not considered while their connections were copied to the incoming words. For example, `University` $\rightarrow$ `or`$\rightarrow$ `Manitoba` became `University` $\rightarrow$ `Manitoba`.

We used two types of graph autoencoders available from [16]: a) regular and b) variational. Both models had the same structure except the final autoencoder followed the VAE structure from [15]. We chose the hidden size in the last graph autoencoder layer as $h = 768$ to make the same size vector similar to the word embedding vector. It is noteworthy that each word in the vocabulary should have a h sized vector from this dependency embedding.

Hence, our embedding scheme had an additional layer containing the dependency information as illustrated in Figure 3. Further normalization on the autoencoder outputs was performed to make them in the same range according to the word embedding numeric range. For unknown words (*i.e.* prepositions), we merely generated a normal random vector for each with the mean and standard deviation from the known word dependency embedding.

3.3. Pretraining and Finetuning Procedure

Before analyzing the results it is important to distinguish the difference between the pretraining and finetuning procedure between our baseline model BERT [7] and ours. BERT used the Wikipedia and BookCorpus to pretrain its 110million+ parameter models as mentioned in Section 2.3. Due to the significant computational and storage requirement we could to attain the same level of pretraining for our unsupervised tasks (EMLM, SDR and GAEs). Furthermore, the pretraining is an expensive task compared to the finetuning ones which are required for the underlying tasks (*i.e.* Classifications, QA *etc*.).

Therefore, we took a different route from BERT mainly due to our resource constraints. We relied on the finetuning procedure rather than large scale pretraining. We used merely *1000 documents* with each less than *256 words* to pretrain our model with the unsupervised EMLM, SDR and GAEs tasks. The reason behind using such small dataset was primarily due to its minimal convergence time. It took around 150k to 200k epochs to reach a negligible loss and almost the 100% accuracy on all three tasks. Furthermore, we initialized our model with BERT's pretrained variables (except newly proposed layers) which ensured somewhat similar performance as BERT into the finetuning phase.

We extended the finetuning (only output layer) by running more epochs compared to BERT due to its reduced memory and computational requirements. For example, the classification and QA tasks had a maximum of 384 tokens

[2]http://universaldependencies.org/docs/en/pos/

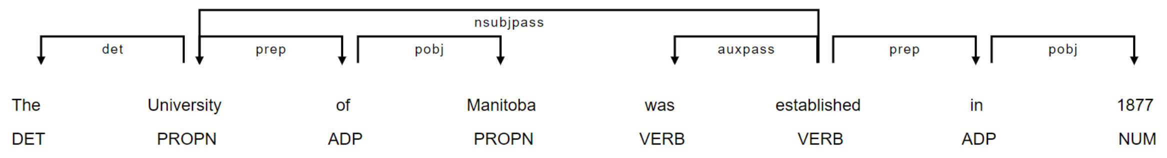


Figure 2: Dependency Parsing of a segment and the connectivity of words generated with Spacy library

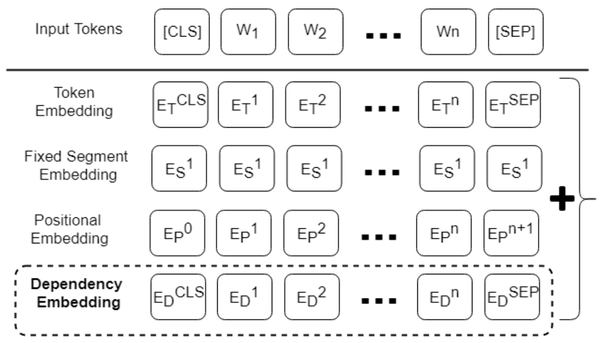


Figure 3: Our Proposed Embedding scheme with additional dependency vector for each word

which we could fit into our GPUs (considering batch size 16/32). Therefore, we ran around 10 to 20 epochs with learning rate (Adam) ranging from $2e-5$ to $3e-5$.

4. Experimental Results

In this section, we detail the targeted datasets briefly and discuss the experimental analysis thereafter. We only provide two Tables (3, 4) concisely displaying our best results. However, throughout the project different performance metrics were collected which is not shows due to space constraints.

4.1. Evaluation Datasets

We use a total of twelve datasets to evaluate our proposed methods and its variations. These datasets can be classified into two categories as we describe them:

Classification: We analyze our classification performance on 8 GLUE datasets [32]:

1. Quora Question Pairs (**QQP**) is a 363k instance binary task with two sentences input to classify their similarity
2. Semantic Textual Similarity Benchmark (**STS-B**) [2] is a similar task (around 6k data) like QQP where two headline similarity is asked; however, the output for the 2 input sentences need to scored numerically between 1 to 5.
3. Microsoft Research Paraphrase Corpus (**MRPC**) [8] is also parallel to the aforementioned tasks where online news headline similarity are measured (binary)
4. Question Natural Language Inference (**QNLI**) [27] is another large two sentence task with 108k instances were given a sentence and question pair, a binary classifier needs to predict whether the answer is in that sentence (0/1).
5. Corpus of Linguistic Acceptability (**CoLA**) [33] contains 8.5k single sentences where some of them are/not linguistically acceptable; hence a binary classifier needs to detect the acceptability.
6. Stanford Sentiment Treebank (**SST-2**) [29] is another single sentence classifier for detecting positive/negative sentiments on movie reviews.
7. Multi-Genre Natural Language Inference (**MNLI**) [34] is a ternary classification task with 392k instances with 2 sentences as inputs.
8. Lastly, we use a much smaller Recognizing Textual Entailment (**RTE**) [9] task with only 2.5k instances with a similar task as MNLI (ternary to binary entailment).

Question/Answering (QA):

1. **SQuAD-v*:** We use two versions (1.1 and 2.0) of Stanford Question Answering Dataset (Stanford Question Answering Dataset (SQuAD) [27] with around 200k question answering pairs. Here, there is an input comprehension of arbitrary length with one/multiple fixed questions. The original answers to the questions were retrieved via crowdsourcing. However, the significant difference between v1.1 and v2.0 is that some questions do not have any answers based on the comprehension in v2.0 compared to v1.1.
2. **TriviaQA:** TriviQA [14] is to the best of our knowledge, the largest QA dataset with 650K instances with questions, answers and a large document.
3. **SWAG:** The Situations With Adversarial Generations (SWAG) is a multiple choice (between 4) QA dataset with 113k instances [36]. Here, the problem can be formulated as a 4-class classifier for the input sentence and the possible entailment sentence.

Table 3: Classification results on GLUE datasets [32] compared with the BERT-base model where maximum sentence length $n \in \{128, 256\}$, learning rate $\in \{2e-5, 3e-5\}$ and batchsize $\in \{16, 32\}$

Method	MNLI	CoLA	QQP	MRPC	QNLI	SST-2	RTE	STS-B	Average
BERT-Base	84.47	83.41	91.38	85.53	91.23	92.32	68.95	89.20	85.81
EMLM	79.86	74.49	90.36	78.67	86.96	88.87	60.29	86.17	80.71
SDR+EMLM	81.8	78.33	90.23	75.98	86.93	90.36	58.84	86.90	81.17
SDR+EMLM+VGAE	79.90	74.21	89.81	73.77	86.56	88.53	60.28	86.17	79.90
SDR+EMLM+GAE	81.38	77.75	90.37	75.24	87.77	88.99	56.67	87.46	80.70

4.2. Implementation Details

To retrofit our unsupervised tasks and GAEs to the original BERT model, we had to modify the existing implementation of BERT [6]. We kept the original structure and pipeline of BERT pretraining and added the new loss functions in Tensorflow for the underlying tasks. The EMLM was introduced while creating the pretraining data where we incorporated another library Spacy for NER. For example, the masking probabilities for each word/token were varied according to Section 3.1.1 in this step. We also prepared the inputs for SDR alongside (Section 3.1.2).

The GAEs (Section 3.2.2) were trained separately (not end-to-end) which is a downside of the whole pretraining step. This is done due to the memory requirement from the 20k vocabulary as it will end up in an adjacent matrix of $20k \times 20k$. Thus, we trained the autoencoder (in a separate program) which took the 1000 input documents and filtered 20k tokens along with their dependency graphs. Afterwards, we generated the 768 sized vector for each token (from $20k$) with GAE/VGAEs and added it with the other embeddings available from BERT. Notably, we ran the GAEs until we converged around 80% accuracy on the unsupervised edge detection task.

The original BERT repository also lacked some of the implementations for the classification and QA tasks. We implemented the missing functions and their evaluations for proper benchmarking. We used the Titan V GPUs with 16/32 GB memory from Amazon EC2 cloud, and the code is readily available at `https://github.com/mominbuet/DLProject.git`.

4.3. Results

In Table 3 and 4, we show our results in classification and QA task respectively along with its comparison with its parental model BERT. Here, we only compare with BERT's base model as the BERT-Large only varied by a few decimal numbers compared to its double size (Table 1). Notably, all these results are performed on the validation set provided from the dataset owners as we did not perform the server validated test where the test set results were manually submitted.

This is partially due to the unimproved results as the classification task results from Table 3 show lower accuracy on all tasks. However, in our ablations, we see that the SDR and EMLM combinedly perform the best with an average of 81.71% compared to BERT's 85.81%. However, it is noteworthy that BERT utilized a much larger pretraining dataset (*i.e.* Wikipedia) with a million iterations compared to our 1000 documents.

Table 4 shows our performance for the aforementioned four QA datasets. Here, the sequence length plays an important role as the answer to any question might be located towards the end of the document. Hence, the results here considered $\{256, 384\}$ sequence lengths which offered a marginal difference in results while consuming larger memory for a longer sequence. We report the best results for all instances of SQuAD whereas for TriviQA we fixed the length to 256.

It is noteworthy that TriviQA contains much longer documents than SQuAD dataset which primarily attributes to the negligible Exact matches and F1 scores. On the other hand, the exact matches for SQuAD ranged between 63.24 to 65.55 for all four of our methods whereas BERT outperformed with 72.95. Needlessly, our approach follows the same criteria for answer span generation according to the original implementation from BERT. The SWAG evaluations are done similarly to the classification where the output layer was transformed into $[768, 4]$ shape.

4.4. Discussions and Future Direction

From the aforementioned results, we see that the classification results (Table 3) are comparatively well rounded throughout all datasets whereas we under-perform for all the QA tasks (Table 4).

This will relate to our earlier choice on relying more on the finetuning rather than pretraining due to our resource constraints. As in finetuning, we train the final output layer from scratch; it was much easier for the shape of $[768, 2]$; whereas a QA task will demand a span of answers $[768, s]$ of length s. Therefore, our finetuning performs much better in the classification task compared to the QA ones.

The impact of GAEs is also minimal due to the smaller pretraining dataset as it limited the connectivity between tokens. However, as there were 20k unique tokens compared

Table 4: Results (Exact Matches/F1 score) on Question Answering datasets with where maximum sequence length $n \in \{256, 384\}$, learning rate $\in \{3e-5, 4e-5\}$ and batchsize $\in \{16, 32\}$

Method	SQuAD-v2	SQuAD-v1.1	SWAG	TriviQA
BERT-Base	72.95/76.36	80.49/88.33	75.45	7.34/8.41
EMLM	65.14/69.15	71.29/81.54	48.46	6/6.83
SDR+EMLM	65.55/69.25	71.83/81.65	47.47	5.64/6.52
SDR+EMLM+VGAE	63.24/67.02	72.38/82.17	50.70	5.74/6.64
SDR+EMLM+GAE	65.38/69.18	70.55/80.8	57.70	6.16/7.41

to the 30k in BERT's vocabulary, our targeted dataset offered only a sparse adjacency matrix. This will change for a more extensive corpus as it will provide more relations and transitions between words, portraying a more holistic view.

Nevertheless, the current results enlighten that even with 1000 documents pretraining, the outputs are somewhat promising for a large scale training involving TPU. Notably, throughout the project, we did not have access to TPUs which we do now via Google Cloud Platform which can benefit if the project proceeds further.

5. Related Works

The prior works in this generalized model for natural language processing (NLP) can be classified as follows:

Word Embedding Vector Generation

Before the deep learning movement and during its early years, a majority of the NLP tasks were attempted using fixed embedding vectors for each word/tokens. This is analogical to the fixed feature detection procedure in Computer Vision (CV) where an arbitrary number of features were extracted from the underlying image, and the higher operations (classification, segmentation *etc.*) were only concerned on these features. Hence, in NLP, these embedding vectors were treated similarly as they played a vital role in the high-level tasks. However, there is one generalizable characteristic to classify these vector generation with —a) Neural Network [3, 22, 30, 21], and b) without Neural network [1, 18, 28].

Around 2010, there were a shift towards using neural networks to generate these vectors as the one-hot encoding alike [1, 28] approaches were only specific for a certain use case. Furthermore, there were efficient un/semi-supervised training procedure proposed with contemporary Neural Network (NN) [30, 21]. Later on, these NN based vectors also showed interoperability on multiple corpora illustrating the word difference/similarity. Furthermore, they were generalized into phrase, sentence, paragraph and even document level embedding [20, 19, 5].

However, these generated embedding vectors for each word in vocabulary are kept same once trained which is currently under modification. Recent works such as ELMo [25] and BERT [7] in 2018 proceed towards a more contextual word embedding. In these methods, the authors proposed the representing vector for each word can be different depending on its usage or placement in a sentence/document. We also pursue the same direction in the work, enhancing the representational ability of these words.

Generalized model for NLP

In 2015 Dai and Le [4] proposed a method to pretrain an LSTM model on a large corpus with two unsupervised tasks (similar to BERT and ours). For example, one of their tasks was the next sentence prediction (binary) related to BERT's pretraining task as discussed in Section 2.3. With the performance improvement from this model, some notable attempts were foreseen in line with this work.

ULMFiT [13] and OpenAI-GPT [26] were the prominent ones as they were proposed in 2018 both of them differ in the underlying neural network scheme— LSTM and Transformer model respectively. However, both have shown improvements from previous works and promises for a generalizable NLP model which is already present in Computer Vision domain.

In summary, these attempts largely inherited the transfer learning properties from CV where knowledge from one task (mostly unsupervised) is transferred towards another supervised task (*e.g.* classification, QA *etc.*) [23, 7].

6. Conclusion

Our proposed approach for constructing a generalized model inherited every feature of the existing state-of-the-art BERT [7] along with some new addition and modification to the original scheme. Our benchmarks on two different NLP tasks show the deficits and potential improvement areas, especially the pretraining dataset. In future, we would like to incorporate a larger corpus while pretraining for the proposed task which should yield better results than the current version. Needlessly, the pipeline and the results discussed in this paper will motivate towards a unified, generalized deep learning model for NLP and prove to be beneficial on transfer learning for different NLP tasks.

References

[1] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[2] 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.

[3] 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. ACM, 2008.

[4] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc., 2015.

[5] Andrew M. Dai, Christopher Olah, and Quoc V. Le. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*, 2015.

[6] Jacob Devlin. TensorFlow code and pre-trained models for BERT. `https://github.com/google-research/bert`, 2008. [Online; accessed 22-April-2019].

[7] 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.

[8] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[9] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007.

[10] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[14] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Thomas N Kipf and Max Welling. Variational graph autoencoders. *arXiv preprint arXiv:1611.07308*, 2016.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[19] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[20] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018.

[21] 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.

[22] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

[23] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*, 2016.

[24] 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.

[25] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237, 2018.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[27] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[28] Holger Schwenk and Jean-Luc Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–765. IEEE, 2002.

[29] 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.

[30] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[31] 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.

[32] Alex Wang, Amapreet 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.

[33] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[34] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[35] 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.

[36] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.

[37] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.