TensorFlow 2.0 Question Answering

Manav Parekh
Stevens Institute of Technology
Hoboken, NJ, USA
mparekh3@stevens.edu

Ameya Swar
Stevens Institute of Technology
Hoboken, NJ, USA
aswar@stevens.edu

Kishan Panchal
Stevens Institute of Technology
Hoboken, NJ, USA
kpancha3@stevens.edu

***Abstract*—Question and Answering systems emulate how people look for information by reading the web to return answers to common questions. Deep learning models can be used to improve the accuracy of these answers. In this project, Our goal is to predict short and long answer responses to real questions about Wikipedia articles. The dataset is provided by Google's Natural Questions. Here we are using the google BERT pre-train model baseline for Tensorflow to create predictions for the Natural Questions test set**

I. Introduction

Question answering (QA) is a well-researched problem in NLP. In spite of being one of the oldest research areas, QA has application in a wide variety of tasks, such as information retrieval and entity extraction. Recently, QA has also been used to develop dialog system and chatbots designed to simulate human conversation. Traditionally, most of the research in this domain used a pipeline of conventional linguistically-based NLP techniques, such as parsing, part-of-speech tagging and co-reference resolution. Many of the state-of-the-art QA systems – for example, IBM Watson use these methods.

However, with recent developments in deep learning, neural network models have shown promise for QA. Although these systems generally involve a smaller learning pipeline, they require a significant amount of training. GRU and LSTM units allow recurrent neural networks (RNNs) to handle the longer texts required for QA. Further improvements – such as attention mechanisms and memory networks – allow the network to focus on the most relevant facts. Such networks provide the current state-of-the-art performance for deep-learning-based QA. In this project, we study the application of several deep learning models to the question answering task. After describing two RNN-based baselines, we focus our attention on end-to-end memory networks, which have provided state-of-the-art results on some QA tasks while being relatively fast to train.

A. Project Goal

For each article + question pair, you must predict / select long and short form answers to the question drawn directly from the article. - A long answer would be a longer section of text that answers the question - several sentences or a paragraph. - A short answer might be a sentence or phrase, or even in some cases a YES/NO. The short answers are always contained within / a subset of one of the plausible long answers. - A given article can (and very often will) allow for both long and short answers, depending on the question.

Github page for the Natural Questions dataset contains helpful utilities and scripts. We are using the simplified text version of the data - most of the HTML tags have been removed, and only those necessary to break up paragraphs / sections are included.

B. Project Flow

File descriptions

- simplified-nq-train.jsonl - the training data, in newline-delimited JSON format.
- simplified-nq-kaggle-test.jsonl - the test data, in newline-delimited JSON format.
- sample-submission.csv - a sample submission file in the correct format.

Data fields

- document-text - the text of the article in question (with some HTML tags to provide document structure). The text can be tokenized by splitting on whitespace.
- question-text - the question to be answered
- long-answer-candidates - a JSON array containing all of the plausible long answers.
- annotations - a JSON array containing all of the correct long + short answers. Only provided for train.
- document-url - the URL for the full article. Provided for informational purposes only. This is NOT the simplified version of the article so indices from this cannot be used directly. The content may also no longer match the html used to generate document-text. Only provided for train.
- example-id - unique ID for the sample.

II. Related Work

Language model pre-training has shown to be effective for improving many natural language processing tasks. Among different models, the most recent google-released Bidirectional Encoder Representations from Transformers (BERT) is a conceptually simple but empirically powerful one. It performs very well in a wide range of tasks, including, Tensorflow

2.0 Question Answering competition.According to the paper, the pre-trained BERT representations can be fine-tuned with additional architectures to succeed in specific tasks.

There is a long history of pre-training general language representations, and we briefly review the most widely-used approaches in this section.

Dataset Description

The Natural Questions (NQ) is a question answering dataset containing 307,373 training examples, 7,830 development examples, and 7,842 test examples. Each example is comprised of a google.com query and a corresponding Wikipedia page. Each Wikipedia page has a passage (or long answer) annotated on the page that answers the question and one or more short spans from the annotated passage containing the actual answer. The long and the short answer annotations can however be empty. If they are both empty, then there is no answer on the page at all. If the long answer annotation is non-empty, but the short answer annotation is empty, then the annotated passage answers the question but no explicit short answer could be found. Finally 0.1 of the documents have a passage annotated with a short answer that is “yes” or “no”, instead of a list of short spans.

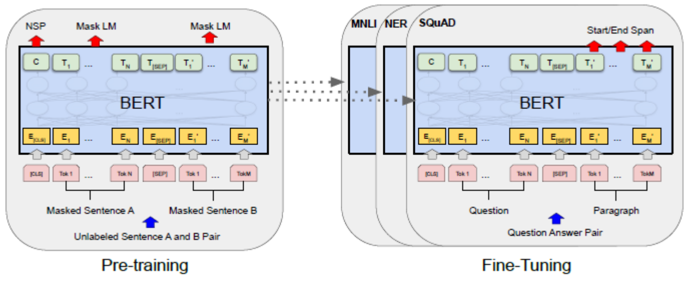
Unsupervised Feature-based Approaches

Learning widely applicable representations of words has been an active area of research for decades, including non-neural (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) and neural (Mikolov et al., 2013; Pennington et al., 2014) methods. Pre-trained word embeddings are an integral part of modern NLP systems, offering significant improvements over embeddings learned from scratch (Turian et al., 2010). To pretrain word embedding vectors, left-to-right language modeling objectives have been used (Mnih and Hinton, 2009), as well as objectives to discriminate correct from incorrect words in left and right context (Mikolov et al., 2013). These approaches have been generalized to coarser granularities, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sentence words given a representation of the previous sentence (Kiros et al., 2015), or denoising autoencoder derived objectives (Hill et al., 2016). ELMo and its predecessor (Peters et al., 2017, 2018a) generalize traditional word embedding research along a different dimension. They extract context-sensitive features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP benchmarks (Peters et al., 2018a) including question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), and named entity recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) proposed learning contextual representations through a task to predict a single word from both left and right context using LSTMs. Similar to ELMo, their model is feature-based and not deeply bidirectional. Fedus et al. (2018) shows that the cloze task can be used to improve the robustness of text generation models.

Unsupervised Fine-tuning Approaches

As with the feature-based approaches, the first works in this direction only pre-trained word embedding parameters from unlabeled text (Collobert and Weston, 2008). More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). The advantage of these approaches is that few parameters need to be learned from scratch. At least partly due to this advantage, OpenAI GPT (Radford et al., 2018) achieved previously state-of-the-art results on many sentencelevel tasks from the GLUE benchmark (Wang et al., 2018a). Left-to-right language modeling and auto-encoder objectives have been used for pre-training such models (Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015).

[h!]



Transfer Learning from Supervised Data

There has also been work showing effective transfer from supervised tasks with large datasets, such as natural language inference (Conneau et al., 2017) and machine translation (McCann et al., 2017). Computer vision research has also demonstrated the importance of transfer learning from large pre-trained models, where an effective recipe is to fine-tune models pre-trained with ImageNet (Deng et al., 2009; Yosinski et al., 2014).

III. Implementation

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Tensorflow 2.0 Question Answering.

BERT’s key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

What Makes BERT Different?

BERT builds upon recent work in pre-training contextual representations — including Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit. However, unlike these previous models, BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus (in this case, Wikipedia).

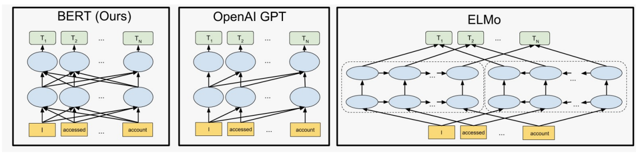
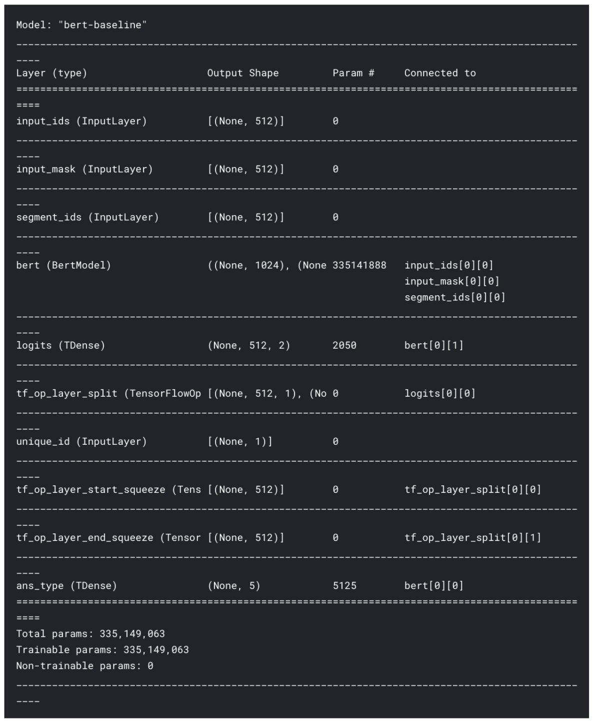


Fig. 1

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section. A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture. Model Architecture BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017) and released in the tensor2tensor library.1 Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani et al. (2017) as well as excellent guides such as "The Annotated Transformer."2 In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.3 We primarily report results on two model sizes: BERTBASE (L=12, H=768, A=12, Total Parameters= 110M) and BERTLARGE (L=24, H=1024, A=16, Total Parameters=340M). BERTBASE was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.4



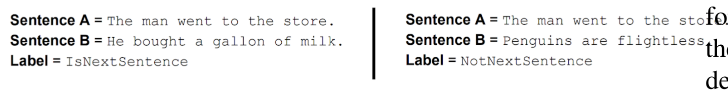
```
Model: "bert-baseline"
----------------------------------------------------------------------------------------------
----
Layer (type)                    Output Shape         Param #     Connected to
==============================================================================================
====
input_ids (InputLayer)          [(None, 512)]        0
----------------------------------------------------------------------------------------------
----
input_mask (InputLayer)         [(None, 512)]        0
----------------------------------------------------------------------------------------------
----
segment_ids (InputLayer)        [(None, 512)]        0
----------------------------------------------------------------------------------------------
----
bert (BertModel)                ((None, 1024), (None 335141888   input_ids[0][0]
                                                                 input_mask[0][0]
                                                                 segment_ids[0][0]
----------------------------------------------------------------------------------------------
----
logits (TDense)                 (None, 512, 2)       2050        bert[0][1]
----------------------------------------------------------------------------------------------
----
tf_op_layer_split (TensorFlowOp [(None, 512, 1), (No 0           logits[0][0]
----------------------------------------------------------------------------------------------
----
unique_id (InputLayer)          [(None, 1)]          0
----------------------------------------------------------------------------------------------
----
tf_op_layer_start_squeeze (Tens [(None, 512)]        0           tf_op_layer_split[0][0]
----------------------------------------------------------------------------------------------
----
tf_op_layer_end_squeeze (Tensor [(None, 512)]        0           tf_op_layer_split[0][1]
----------------------------------------------------------------------------------------------
----
ans_type (TDense)               (None, 5)            5125        bert[0][0]
==============================================================================================
====
Total params: 335,149,063
Trainable params: 335,149,063
Non-trainable params: 0
----------------------------------------------------------------------------------------------
----
```

Fig. 2: BERT baseline Architecture

Pre-trained representations can either be context-free or contextual, and contextual representations can further be unidirectional or bidirectional. Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary. For example, the word "bank" would have the same context-free representation in "bank account" and "bank of the river." Contextual models instead generate a representation of each word that is based on the other words in the sentence. For example, in the sentence "I accessed the bank account," a unidirectional contextual model would represent "bank" based on "I accessed the" but not "account." However, BERT represents "bank" using both its previous and next context — "I accessed the ... account" — starting from the very bottom of a deep neural network, making it deeply bidirectional.

The Strength of Bidirectionality

If bidirectionality is so powerful, why hasn't it been done before? To understand why, consider that unidirectional models are efficiently trained by predicting each word conditioned on the previous words in the sentence. However, it is not possible to train bidirectional models by simply conditioning each word on its previous and next words, since this would allow the word that's being predicted to indirectly "see itself" in a multi-layer model. To solve this problem, we use the straightforward technique of masking out some of the words in the input and then condition each word bidirectionally to predict the masked words. For example:

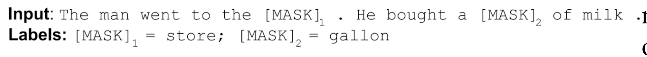


Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the sto
Sentence B = Penguins are flightless
Label = NotNextSentence

Fig. 3

While this idea has been around for a very long time, BERT is the first time it was successfully used to pre-train a deep neural network. BERT also learns to model relationships between sentences by pre-training on a very simple task that can be generated from any text corpus: Given two sentences A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence? For example:



Input: The man went to the [MASK]$_1$. He bought a [MASK]$_2$ of milk .
Labels: [MASK]$_1$ = store; [MASK]$_2$ = gallon

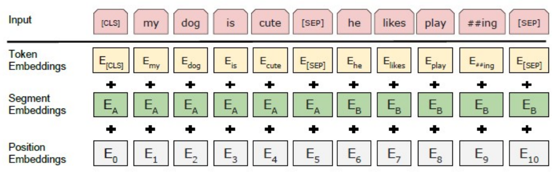
Fig. 4

Input/Output Representations To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., h Question, Answer i) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embedding to every token indicating whether it belongs to sentence A or sentence B. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special [CLS] token as C 2 RH, and the final hidden vector for the ith input token as Ti 2 RH. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. A visualization of this construction can be seen in Figure 2.

Pre-training BERT

Unlike Peters et al. (2018a) and Radford et al. (2018), we do not use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, we pre-train BERT using two unsupervised tasks, described in this section. This step is presented in the left part of Figure 1.

Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-toright and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context. former is often referred to as a "Transformer encoder" while the left-context-only version is referred to as a "Transformer decoder" since it can be used for text generation. In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a "masked LM" (MLM), although it is often referred to as a Cloze task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input. Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, we do not always replace "masked" words with the actual [MASK] token. The training data generator chooses 15prediction. If the i-th token is chosen, we replace the i-th token with (1) the [MASK] token 80the time (2) a random token 10the unchanged i-th token 10Ti will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix C.2.


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

4.jpg

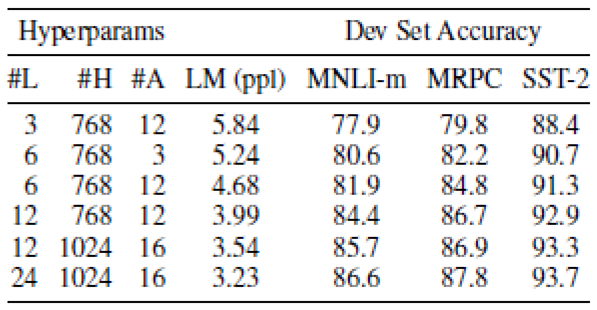
Fig. 5: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Fine-tuning BERT

Fine-tuning is straightforward since the selfattention mechanism in the Transformer allows BERT to model many downstream tasks— whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs. For applications involving text pairs, a common pattern is to independently encode text pairs before applying bidirectional cross attention, such as Parikh et al. (2016); Seo et al. (2017). BERT instead uses the self-attention mechanism to unify these two stages, as encoding a concatenated text pair with self-attention effectively includes bidirectional cross attention between two sentences. For each task, we simply plug in the taskspecific inputs and outputs into BERT and finetune all the parameters end-to-end. At the input, sentence A and sentence B from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text-? pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for tokenlevel tasks, such as sequence tagging or question answering, and the [CLS] representation is fed into an output layer for classification, such as entailment or sentiment analysis. Compared to pre-training, fine-tuning is relatively inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model.

Feature-based Approach with BERT

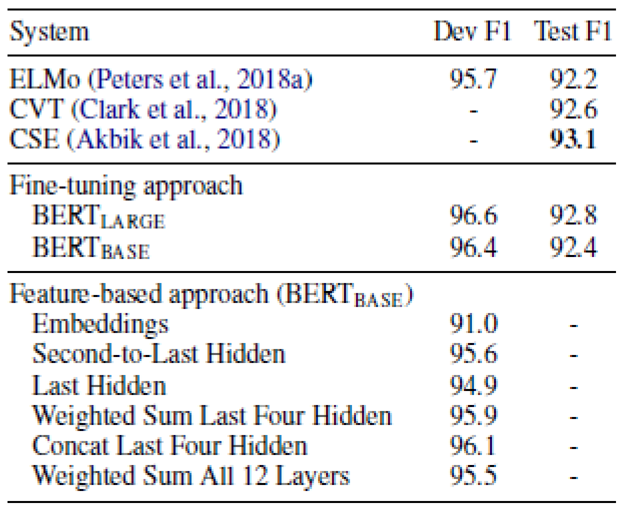
All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the feature-based approach, where fixed features are extracted from the pre-trained model, has certain advantages. First, not all tasks can be easily represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added. Second, there are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation. In this section, we compare the two approaches by applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task (Tjong Kim Sang and De Meulder, 2003). In the input to BERT, we use a case-preserving WordPiece model, and we include the maximal document context provided by the data. Following standard practice, we formulate this as a tagging task but do not use a CRF



Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Fig. 6: Ablation over BERT model size. L = the number of layers; H = hidden size; A = number of attention heads. “LM (ppl)” is the masked LM perplexity of held-out training data.

layer in the output. We use the representation of the first sub-token as the input to the token-level classifier over the NER label set. To ablate the fine-tuning approach, we apply



System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
BERT$_{LARGE}$	96.6	92.8
BERT$_{BASE}$	96.4	92.4
Feature-based approach (BERT$_{BASE}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Fig. 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

the feature-based approach by extracting the activations from one or more layers without fine-tuning any parameters of BERT. These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer. Results are presented in Table 7. BERTLARGE performs competitively with state-of-the-art methods. The best performing method concatenates the token representations from the top four hidden layers of the pre-trained Transformer, which is only 0.3 F1 behind fine-tuning the entire model. This demonstrates that BERT is effective for both finetuning and feature-based approaches.

IV. RESULTS

By implementing pre-trained BERT model we can achieve 0.7 accuracy. We can further improve the accuracy by fine-tuning the parameters.

V. CONCLUSION

Recent empirical improvements due to transfer learning with language models have demonstrated that rich, unsupervised pre-training is an integral part of many Question Answer systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep bidirectional architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.

VI. FUTURE SCOPE

Implementing different architecture like QRN, EntNet, DMN+. For other dataset like SQuAD2.0 this methods has given better results.

VII. REFERENCE

https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

https://towardsdatascience.com/how-does-bert-reason-54feb363211

https://web.stanford.edu/class/cs224n/reports/default/15848021.pdf

https://github.com/google-research/language/tree/master/language/question$_a nswering/bert_j oint$

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/