IRE assignment 3

Report

1. ## What is the problem that the authors are trying to solve and why is it significant?

   Search engines have been trained to use unigrams matching to give relevant results for a search query. Due to this feature of a search engine, users are also trained to just put keyword queries without giving any heed to the syntax or grammar of the queries themselves. However, recently there has been an uproar of verbose queries owning to the popularity of voice-based searching (like Alexa, ok Google etc). Deep learning algorithms have provided a way to process such verbose queries effectively. But, these verbose queries have been shown to exhibit a lack of coherent structure and violate grammar rules. The authors in the paper are trying to automatically determine whether a search query is well-formed or not. This classification would help various downstream tasks like understanding the user's intent and generating better query related suggestions in search engines.

2. ## What is transfer learning?

   Transfer learning was first introduced in computer vision tasks. Transfer learning is a machine learning technique that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. It gained popularity as it significantly reduced the amount of time and resources required to train a model from scratch and still gave promising results. Transfer learning only works in deep learning if the model features learned from the first task are general. Transfer learning has been applied to cancer subtype discovery, text classification, digit recognition, medical imaging and spam filtering. Since the model's task is general transfer learning also helps in understanding and quickly iterating over models to get better results.

3. ## Explain the architecture of the paper in your own words.

   The architecture of the model utilizes an inductive transfer learning approach by using a language model that is already pre-trained on a large corpus. They have utilized the state-of-the-art Averaged-SGD Weight-Dropped Long Short Term Memory (AWD-LSTM) networks. Their variant is a simple LSTM with no short-cut connections, no attention or any advance mechanisms, with the same hyperparameters as in typical LSTMs with the addition of tuned drop-connect hyperparameters.
   The architecture model is divided into three phases:

Phase 1: General Domain Pretraining

The authors selected a language model which was pretrained on the huge English corpus of WikiText-103 dataset containing 103 million unique words and 28,595 preprocessed Wikipedia articles. This helped in learning the general language structure and dependencies of English language.

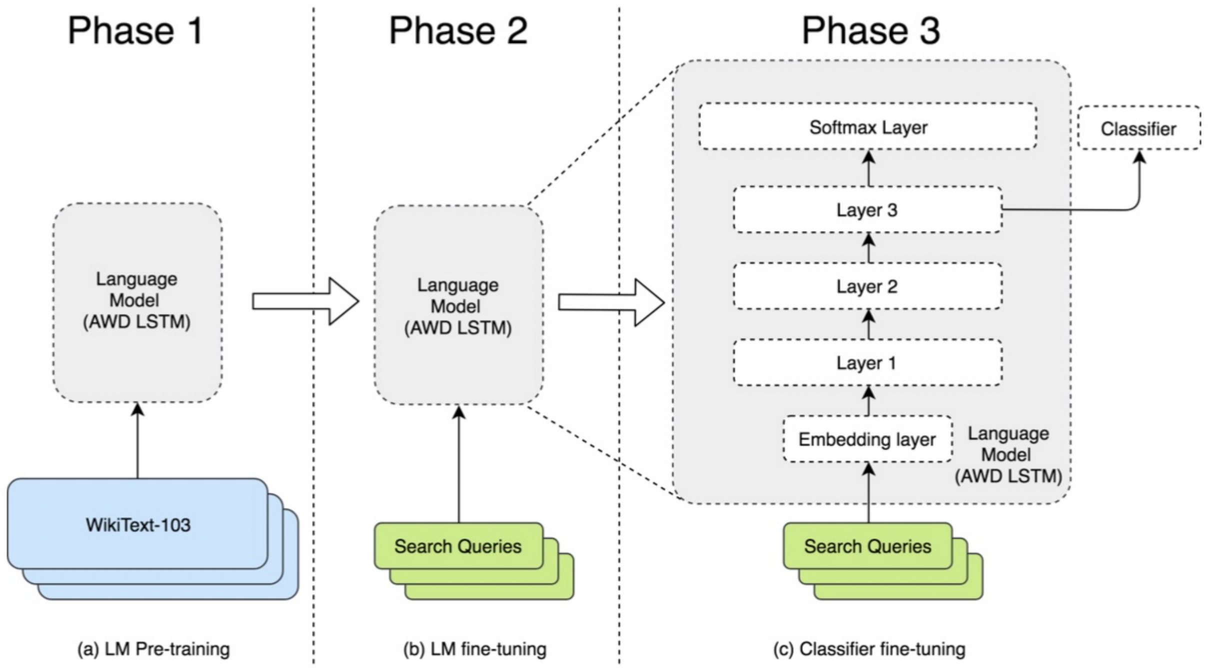
Phase 2 : Language model Fine-tuning for a specific task

The data present for a specific task generally comes from a specific distribution. As the architecture's aim is to capture this distribution, fine tuning on this distribution is essential for the language model. To achieve this, the authors used task-specific data to fine-tune the model in an unsupervised manner. They used two methods to combat catastrophic forgetting:

1. Discriminative Fine-tuning (*DFT*): For each of the three different layers in the AWD-LSTM model, a different learning rate was used. The intuition behind this was that since each of the layers represent a different kind of information, they must be fine-tuned to different extents.
2. Slanted Triangular Learning Rates (*SRLR*): To help the model converge to the desired parameter space, the authors changed the learning rate during tuning as the number of training samples increased. A slanted triangular learning rate method was used where first the learning rate was increased and then linearly decayed as the number of training samples increases.

Phase 3: Classifier Fine-tuning for the target Task

The upstream architecture of the language model had be kept same but has been appended by two fully connected layers for the final classification, keeping the last layer to predict the well-formedness rating. The weights obtained from the second phase were fine tuned on this architecture.
They have employed Gradual Unfreezing to curb fine-tuning of all the layers at the same time. The model first unfreezes the last layer and trains for one epoch. Subsequently, the next frozen layer is unfrozen and all unfrozen layers are fine-tuned. This is repeated until all layers are fine-tuned until convergence is reached.



4. Discuss and describe the dataset

The dataset contains 25,100 queries from the parallax corpus (Fader et al., 2013). Every query was annotated by five raters each with 1/0 rating of whether or not the query is well-formed. A query is annotated as well-formed if the supplied query is grammatical in nature, has perfect spellings and is an explicit question. For each query, there is a field named score which is the average of the 5 binary judgements. A few examples are:

Query	Wellformedness rating
Which form of government is still in place in greece ?	1.0
Population of owls just in north america ?	0.0
Is johnny depp a celtic fan ?	0.8

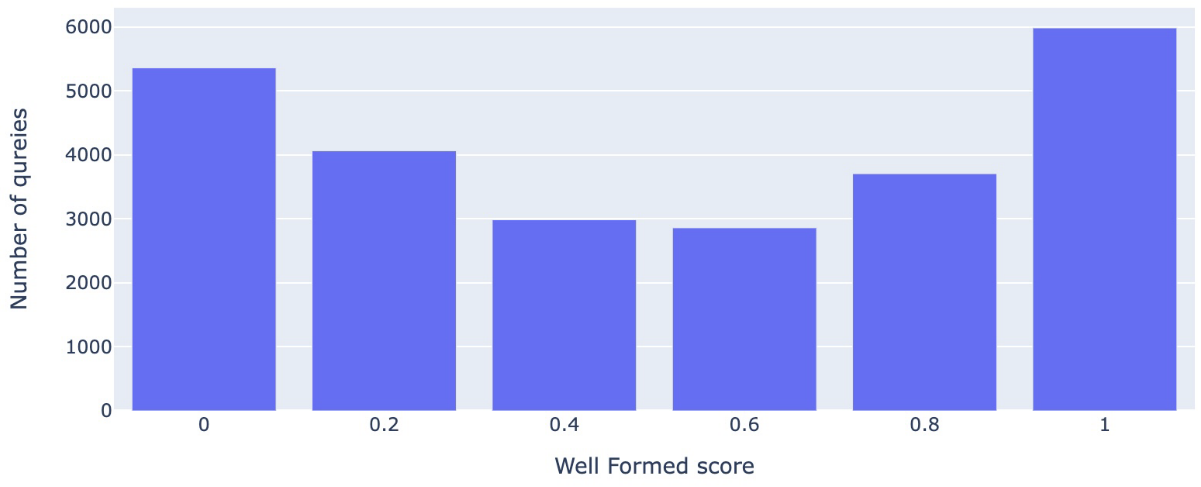
There are three files available:

File	No. of queries
train.tsv	17,500
dev.tsv	3,750

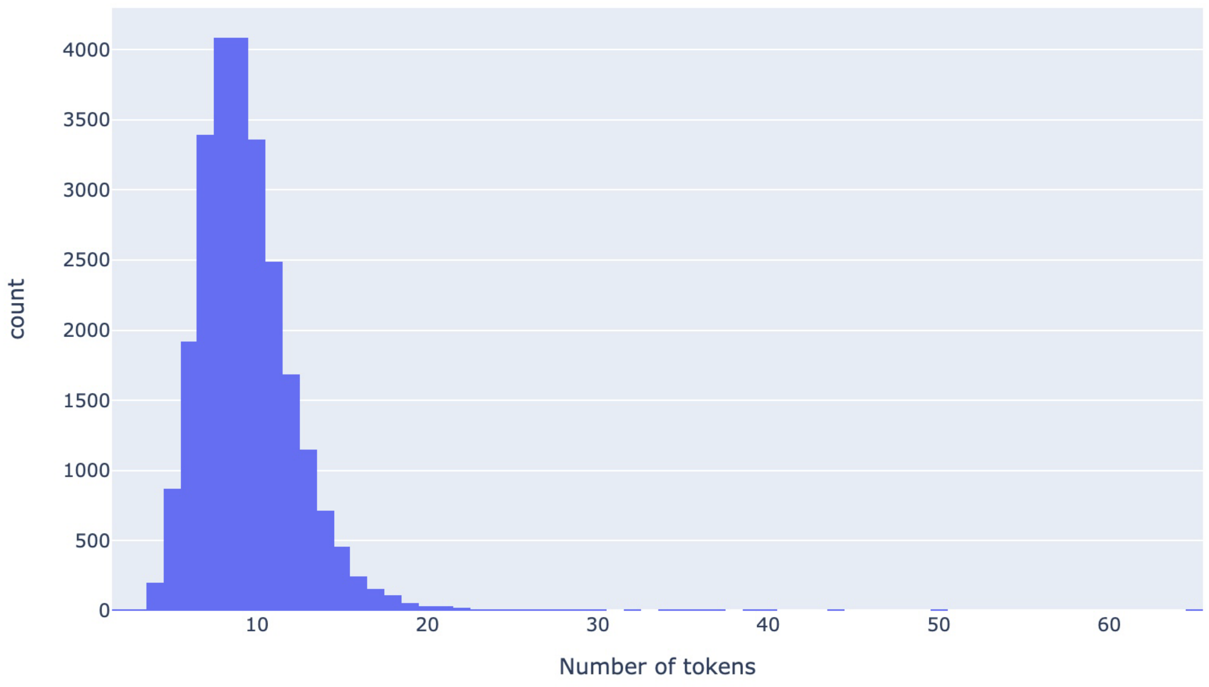
test.tsv	3,850

Some analysis on the dataset:

Class/score distribution overall dataset



Number of tokens using BertTokenizerFast in each query overall dataset



5. Describe the results and takeaways from the paper.

   The overall ITL model has an accuracy of 75.03% improving by a margin of 5% from the state-of-the-art technologies in use (word-1,2 char-3,4 POS-1,2,3 grams: 70.2%). To assess the impact of the three phases, an ablation study was performed:
   - No pretraining: Train the model without the pretraining step: Accuracy 68.2%
   - No LM fine-tuning: Phase 2 is ignored: Accuracy 72.8%
   - Fine-tuning without DFT and STLR: Accuracy 73.0%
   - No gradual unfreezing: Accuracy 72.4%
   - Combined: 75.03%

   We can observe that all the three steps of fine tuning LM, using DFT and STLR and gradual unfreezing help contribute towards improving accuracy.
   The key takeaways include how each step of the process is beneficial and how the ITL model beats the baseline model by a significant margin.

6. BERT Model and fine-tuning

I used two approaches to fine-tune the BERT model.

a. Explicitly defining every appended layer on the BERT model
   For our binary classification, the given dataset has been converted into a 0/1 label by considering queries having a score more than or equal to 0.8 as a well formed query (label 1).
   The hyperparameters tuned are:
   1. Since most of the text in the training dataset had 25 or less tokens, max_seq_length was sent to 25 for encoding.
   2. Batch size: 32
   3. Two fully connected layers with (768, 523) and (512, 2) dimensions with a ReLU activation layer and a dropout layer with probability 0.1 between the two layers.
   4. AdamW optimizer with a learning rate of $1e^{-4}$
   5. Cross entropy loss functions
   6. Epochs: 3
   7. Storing the best validation loss across all the epochs

   On an average I got an accuracy of 73-75% if bert-base-uncased was used, and an accuracy 69-70% for bert-base-cased.
b. Using bert-sklearn library
   Here I merged the train and dev set as the library itself breaks the training set into train and validation dataset.
   The hyperparameters tuned are:
   1. Number of hidden layers having a dimension of (768, 500): num_mlp_layers=1
   2. Max sequence length to consider for encoding: max_seq_length=64

On an average I got an accuracy of 81.1% for bert-base-uncased.
The advantage of the first method is that it trains in under 5 minutes and still gives and accuracy of 75% whereas the library takes 28 minutes to train but it gives an higher accuracy of 81%.

Code link:
Method 1:
https://colab.research.google.com/drive/1JeOBQcyoDqCcgwWyffYdddFLxwWZiuWV?usp=sharing

Method 2:
https://colab.research.google.com/drive/1r0UUIuYQFot_8_q9ayVv8CL0_k76J2P4?usp=sharing

7. In this paper, we understood the task of binary prediction for query well-formedness and finetuned a BERT model on the same task. Now, how can the same architecture be extended to predict the rating of the query?

In my method 1, since initially our output was a 2 length vector scores of the two labels, we can extend the architecture by increasing the output dimension from 2 to 6, and apply Log soft max to get the probability of all 6 possible labels. Then the label scoring the maximum can be given as the predicted answer. Since the cross entropy function can handle multiple probabilities, the loss function need not be changed.