IRE Assignment 3

Name: Triansh Sharma
Roll No.: 2019101006
Batch: UG3 CSE

Answer 1

Nowadays, due to the development of voice-based search mechanisms, most of the queries written on search engines are verbose and not well-formed. This lack of well-formedness in the query might adversely impact the downstream pipeline responsible for processing these queries. Many verbose queries do not follow a coherent structure and may also violate grammatical rules, which enforce tailor-made processing of these queries. This becomes a challenging task where we need to extract relevant information from the query and understand the user's intention. This is the main task the authors are trying to address is by classifying queries into two distinct classes - Queries that are well-formed and that are not, which will aid in various downstream tasks like understanding the user's intent and generating better-related query suggestions in search engines.

Answer 2

Transfer learning is a technique in deep learning that focuses on storing knowledge gained from a previously used machine learning model while solving one problem and applying it to a different related problem. The model developed for a previous task is reused as a starting point for another related study. The technique requires vast computing power and a significant amount of time to train. This type of learning is prevalent in Computer vision and NLP tasks since they need us to train large neural networks, and transfer learning helps them boost their performance by providing the necessary extra skill. Transfer learning goes hand in hand with Domain adaption techniques. We further pre-train a previously built model to the domain of a target task to achieve better results when validated on the task-specific dataset.

Answer 3

The architecture of the proposed model uses inductive transfer learning further divided into three different phases. The authors used the state-of-the-art Averaged-SGD Weight-Dropped Long Short Term Memory (AWD-LSTM) networks as the desired model for learning. The AWD-LSTM language model contained three layers, 1150 hidden activations per layer and an embedding size of 400. The hidden layer of the classifier is of size 50. A batch size of 30 is used to train the model. The LM and classifier fine-tuning is done with a base learning rate of 0.004 and 0.01, respectively.

Phase 1 (Pretraining)

The first phase of the approach involved pretraining the LM on a vast English corpus (WikiText-103 dataset), which comprised nearly 103 Million unique words and 28,595 preprocessed Wikipedia articles. The step imparted general language dependencies to the model before the fine-tuning with task-specific data began.

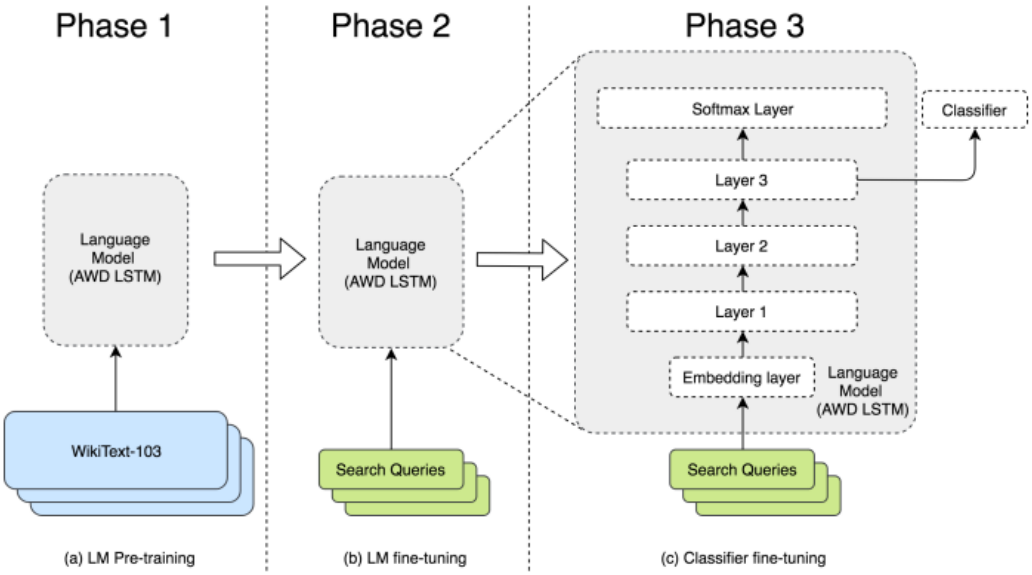


Fig. 1. Inductive Transfer Learning mechanism to identify well-formed search queries

Phase 2 (LM Fine-tuning)

Now the authors used task-specific data to fine-tune their LM in an unsupervised manner. For fine-tuning, they used two strategies, Discriminative Fine-tuning (DFT) and Slanted Triangular Learning Rates (STLR). The purpose of DFT was to use different learning rates to fine-tune the three layers of the AWD-LSTM model to varying extents since each layer represented a diverse variety of information. In SLTR, the authors used a slanted triangular learning rate which first increases and decays linearly as training samples grow. SLTR helped the model to converge to a suitable region of the parameter space.

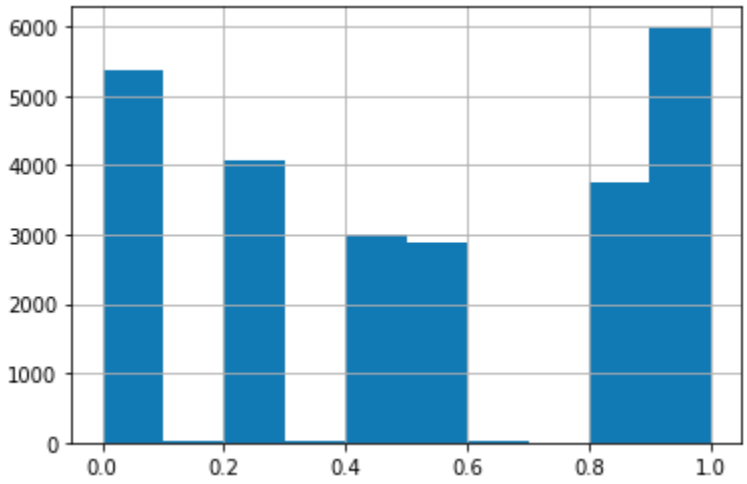
Phase 3 (Classifier Fine-tuning)

In the final phase, the authors appended two fully connected layers for final classification, the last layer responsible for calculating the well-formedness rating. The query was considered well-formed if the well-formedness rating obtained was greater than equal to 0.8. They used a gradual unfreezing heuristic for this task. Gradual unfreezing is a method in which the model is gradually unfrozen, starting from the last layer. This is done to incorporate the fact that the last layer will contain the least amount of information. It is fine-tuned for the first epoch, and the process is repeated until all layers are fine-tuned

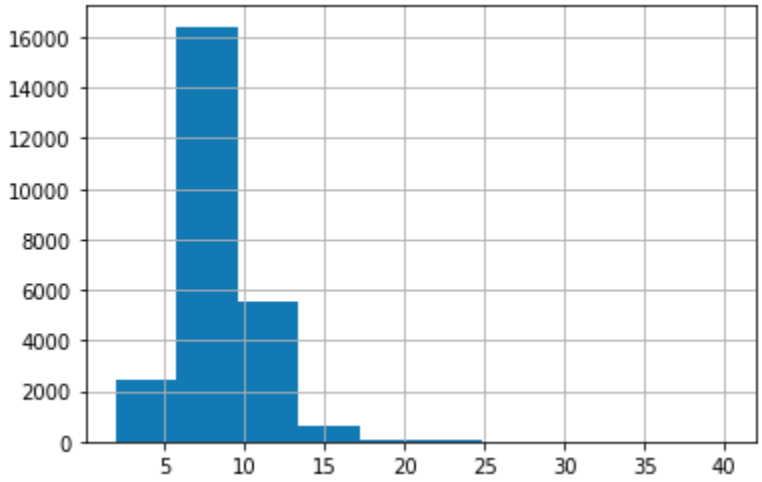
and reach convergence. In other words, the next frozen layer is unfrozen, and all unfrozen layers are fine-tuned.

Answer 4

The dataset comprised in total of 25,100 questions. The rating of queries was labelled between 0 and 1. The authors collected these questions by utilising questions asked by users on WikiAnswers, published initially as the Parallex corpus. The dataset comprises both well-formed queries as well as typical search queries. For experiments performed by the author, they split the dataset into a standard train-dev-test split, consisting of 17500 training, 3750 development, and 3850 test queries. A query is annotated as well-formed in the dataset if the supplied query is grammatical, has correct spellings, and is an explicit question. The combined rating is an average score over ratings given by five different annotators. Apart from what is described in the paper, the dataset has the **distribution of ratings.**



The **distribution of the number of words** in each sentence is described below. Word tokenisation is done using nltk.tokenize.word_tokenize.



Answer 5

The authors achieved an accuracy of 75.03%, improving by nearly five absolute percentage points over the state-of-the-art models. They performed ablation studies with No pretraining, No LM fine-tuning, Fine-tuning without DFT and STLR and No gradual unfreezing. In either of the case, the accuracy of the model decreased. Thus they concluded that DFT and SLTR were beneficial for the model, and gradual unfreezing increased the model's performance; thereby, each phase contributed an increased accuracy. This paper demonstrated the importance of further pretraining the model for domain adaption and learning general language dependencies. Pretraining must be done with considerably large corpora to gain their advantages. The paper also described how learning rates (hyperparameters in general) could be modified to obtain better convergence regions during training iterations. At last, heuristics play an essential role in determining the performance of models.
The below table describes the results of experiments and ablation studies conducted by the authors of this research paper.

Table 2. Comparison of Various Classifiers and Ablation Study for the ITL Model

Model	Accuracy (%)
Question Word Classifier	54.9
Majority Class Prediction	61.5
Word BiLSTM Classifier	65.8
word-1,2 char-3,4 grams [8]	66.9
word-1,2 POS-1,2,3 grams [8]	70.7
word-1,2 char-3,4 POS-1,2,3 grams [8]	70.2
.....	.....
(Inductive Transfer Learning)	
No pretraining with WikiText-103	68.2
No LM fine-tuning	72.8
Fine-tuning without DFT and STLR	73.0
No gradual unfreezing	72.4
All (Pre-train + Fine-tune with DFT and STLR + Gradual unfreezing)	**75.0**

Answer 6

I have used the hugging face transformers library to use a pre-trained BERT model. The fine-tuned model was **trained on combined training and development set (train.tsv + dev.tsv), and the test set (test.tsv) was used as a validation set**. In total, the model was able to achieve **82.44%** accuracy on the validation set.

Hyperparameters, Loss function and Optimizers

1. Maximum Length of tokenized vector = 32
2. Learning rate = 1e-5
3. Train batch size = 32
4. Validation batch size = 8
5. Epochs = 2
6. Optimizer = Adam
7. Loss function = BCE with logit loss

Along with a pre-trained BERT model, an additional linear pre-classifier with **512 neurons** was added with a dropout of 0.3 and a ReLU activation function. The output from this is passed to a classifier, which outputs a single value. When calculating accuracy, the sigmoid function is used to convert it into probabilities, and if the probability is greater than 0.5, then the sample is labelled as 1 else 0.

Answer 6 (Bonus)

For further pretraining, I have used the BertForMaskedLM model from the transformers library. It was further trained on the part of the WikiText-103 Dataset. Since I don't have an Ada account, there were restrictions posed by Google Colab on GPU Usage, time and Runtime environment; thus, I could not perform pretraining over the whole dataset. The pretraining of the model was done in multiple iterations where the model was saved after training some data and then loaded to train on more data.

Hyperparameters and Optimizers for pretraining

1. Maximum Length of tokenized vector = 32
2. Learning rate = 2e-6
3. Batch size = 128
4. Optimizer = Adam
5. Masking Probability = 0.15
6. Epochs = 1

Since Google Colab doesn't allow very high usage of GPU at once, I downloaded the model after each pretraining (done thrice), uploaded its archive to one drive, downloaded the zip folder in Colab from one drive and then loaded the model for fine-tuning(Uploading speed in Colab is very slow). You can check the progress bar in the bonus notebook.

You can download the word level dataset of WikiText103 from here.
Size of WikiText-103 dataset used = **187.6 MB** (**34%** of the original)
The total accuracy achieved after finetuning was **61.55%**.
The dataset used comprised **300K** paragraphs which nearly correspond to **~1.75M** sentences.

Although there might be many reasons for such a low accuracy, one might be the lack of pretraining data since only 34% of the dataset was used. The research paper also described heuristics and modifying learning techniques (SLTR/DFT) to improve performance, although they were not implemented in the code; hence the learning might be poorer.

Hyperparameters, Loss function, Schedulers and Optimizers for finetuning

1. Maximum Length of tokenized vector = 32
2. Learning rate = 1e-3
3. Train batch size = 64

4. Validation batch size = 64
5. Epochs = 10
6. Optimizer = SGD
7. Loss function = NNLoss
8. Scheduler = Exponential LR
9. Scheduler gamma = 0.9

Along with a pre-trained BERT model, an additional linear pre-classifier with **512 neurons** was added with a dropout of **0.5** and a ReLU activation function. Instead of outputting a single value like the previous, the classifier returns two outputs (classes). These two outputs are passed to a LogSoftMax layer which gives the probability of classes, argmax of the output is produced as the label.

Answer 7

Since we have to predict the rating in intervals of 0.2 between zero and one, there are only six possibilities.

0.0 , 0.2 , 0.4 , 0.6 , 0.8 , 1.0

Instead of directly moving into a regression-based model, we can treat the problem by having six different classes change the BERT architecture accordingly. The changes involved would be

1. Since the number of classes is no longer binary, the classification layer in our model should output a 6-D vector since we have six categories.
2. The outputs should be probabilities; we can apply an extra SoftMax (or LogSoftMax) layer in it.
3. In the training loop, we should consider the index with max probability as our prediction, so we will require an extra dictionary where indices are mapped to their ratings.
4. The loss function will no longer be binary cross-entropy (BCE). We can use cross-entropy or any other loss function for classification as long as it does not restrict the number of classes.