Question Classification and Answering System Android VS IOS Dataset

Sarah Hussien 49-16011
Hams Wael 49-4485

May 19, 2024

1 Introduction and Motivation

Natural language processing (NLP) question answering and classification systems are at the forefront of revolutionizing how machines comprehend and react to human language. These systems are made to classify and analyze user queries so that computers can respond with relevant and precise answers. Question answering and classification systems are vital for improving the efficacy and efficiency of information retrieval processes in a variety of applications, including virtual assistants, search engines, and customer support platforms. They achieve this by utilizing sophisticated algorithms and linguistic analysis.

The growing need for intelligent and user-friendly technology that can efficiently interpret and respond to human language is the driving force behind the advancement of question classification and answering systems in NLP. Systems that can quickly and accurately comprehend user queries and deliver insightful answers are becoming more and more necessary in the current digital era, where information overload is an increasingly common issue. We can speed up information retrieval procedures, improve user experiences, and enable more effective human-machine communication by increasing the precision and speed of question classification and answering systems.

Understanding how question classification and answering work as an NLP task is of importance for several reasons. It makes it possible for developers to create more reliable and accurate models by understanding the underlying techniques and algorithms. They can improve the performance and dependability of question classification and answering systems by making well-informed decisions about model architectures, feature representations, and training methodologies.

2 Literature Review

This section will be tackling recent works related to the NLP task "Question Classification and Answering" This section is structured to address the two main steps of this task: Question Classification and Question Answering.

2.1 Question Classification

Question classification is an important task in Natural Language Processing (NLP) that involves categorizing questions into distinct classes or categories based on their semantic meaning and intended answer type. The field of question classification has advanced significantly with the introduction of machine learning and deep learning techniques. On datasets of labeled questions, supervised learning algorithms like random forests and Support Vector Machines (SVM) have been used to train models. These models categorize unseen questions into predefined categories by using patterns and features they have learned from the training data. According to [AD23], contextual and semantic information about questions has been recently captured by deep learning models like neural networks and

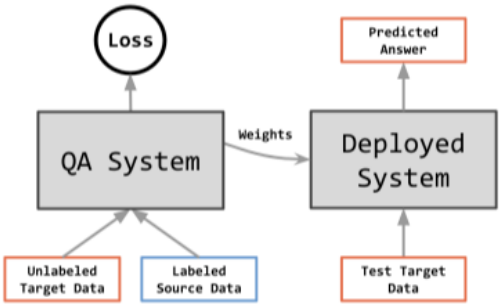

Figure 1: Question Answering Model [YZK+22].

transformer-based models like BERT, allowing for more precise classification.

Question Classification has two main approaches: Binary Classification and Multi-class Classification. Classifying questions into two different classes or categories is known as binary question classification. The main goal is to identify which of the two categories a given question belongs in. While, classifying questions into more than two different classes or categories is known as multi-class classification. Based on the question's domain, intended answer type, or semantic meaning, each one is assigned to a particular class. Multi-class classification allows the system to respond to different kinds of questions with greater accuracy and specificity depending on the category that it has identified.

Question classification can be done in two ways: automatically and manually [RSJ10]. Machine learning or deep learning models trained on labeled question datasets are used to classify questions automatically. These models categorize new, unseen questions into relevant categories by using the patterns and features found in the training data. On the other hand, manual question classification classifies questions by human intervention.

2.2 Question Answering

In order to seamlessly bridge the gap between classifying questions and delivering precise answers, question answering systems are essential. After classifying questions, these systems use a range of methods and algorithms to extract and combine pertinent data to produce thorough responses. According to [YZK+22], a model is trained with labeled source data and unlabeled target data. The resulting system is deployed to answer target questions.As shown in Fig.1: Question Answering Model, [YZK+22] provides a simple illustration of a Question Answering model.

Question Answering (QA) systems have become extremely efficient instruments for automatically providing natural language answers to queries posed by humans. These systems can use a pre-structured database or a wide range of natural language documents. Consider QA systems an advanced form of information retrieval. The demand for this kind of system increases on a daily basis since it delivers short, precise and question-specific answers [SP20]. According to [AH12], QA is made up of three separate modules, each of which comprises a core component in addition to other supporting elements. Question classification, information retrieval, and answer extraction are these three essential elements. By categorizing submitted questions based on their type, question classification plays a crucial part in QA systems. Information retrieval is crucial to answering questions because if no correct answers are present in a document, no further processing could be carried out to find an answer. Ultimately, the goal of answer extraction is to obtain the response to a query posed by the user [AH12].

According to [SP20], prior research primarily characterized the architecture of question-answering systems into three macro-modules, as depicted in Fig.2: question processing, document processing, and answer processing.

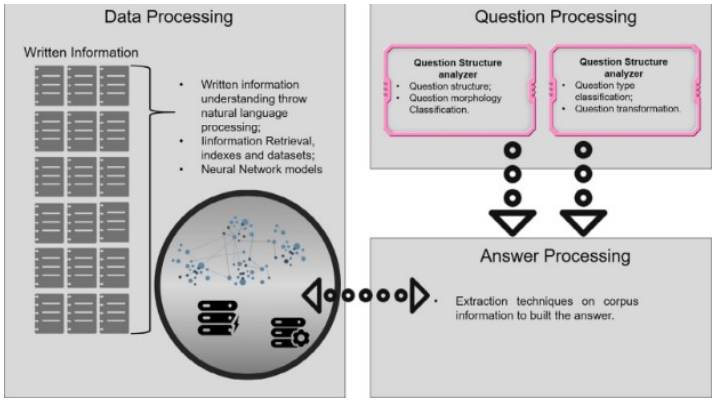

Figure 2: Architecture of the three macro question answering modules [SP20].

Question processing module classifies the question by its type and morphology. Answer processing module uses the classification and transformation made in question processing module to extract the answer from the result of the Document Processing module that executes previously to create datasets, indexes or neural models [SP20].

The user submits a query in natural language for question processing to analyze and categorize. The purpose of the analysis is to determine the question's type, or its main focus. To prevent confusion in the response, this is essential [MSB13].

There are two primary steps involved in processing questions. Analyzing the question's structure is the first step. The second step is to transform the question into a meaningful formula that fits within the domain of QA [HAA16]. The expected kind of response can also serve as a definition for a question. Factoids, lists, definitions, and difficult questions are among the categories [KM11].

Document processing is different from question processing in that it selects a set of relevant documents and extracts a set of paragraphs based on the question's focus or text understanding through natural language processing [MSB13]. Question processing executes on every question posed by the user. The source for the answer extraction can come from this task, which can produce a neural model or a dataset. Ranking the retrieved data based on how relevant it is to the query is possible [NL15].

The most challenging aspect of a question-answering system is the answer processing. This module presents a solution based on extraction techniques applied to the output of the Document Processing module. The response must be straightforward and address the topic; nonetheless, it may necessitate summarizing, combining data from several sources, or handling ambiguity or contradiction [SP20].

2.3 Pretrained Models used in Question Answering and Classification Task

In question answering and classification tasks, several natural language processing (NLP) approaches can be employed to effectively process and understand text data.

According to [ZLC+23], ALBERT, a more efficient and compact variant of the original BERT model, is used to build a QA system. It is based on the Transformer architecture. Its improvement on the BERT model mainly includes the following three aspects: embedded layer decomposition, cross-layer parameter sharing, and SOP sentence order prediction task.

Another model used in QA tasks is BamnetTL - Bidirectional Attention Memory Network with Transfer Learning. According to [SGW+23], BamnetTL is used in is Q and A matching, wherein the appropriate answer is chosen from candidate responses. It incorporates deep feature transfer based on a bidirectional attention memory network. //

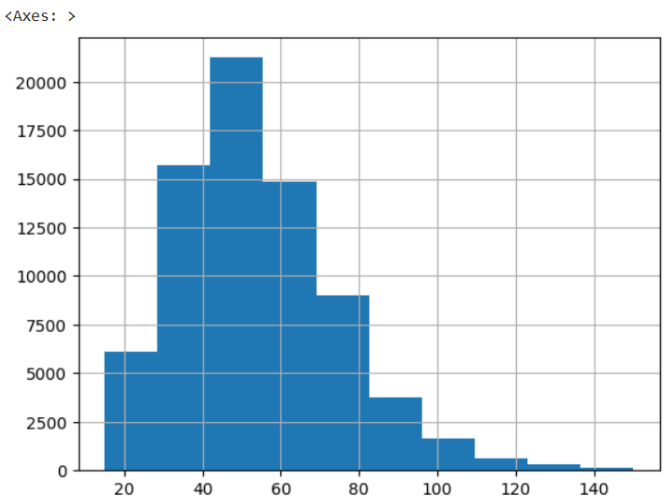

Figure 3: Title Distribution of question lengths.

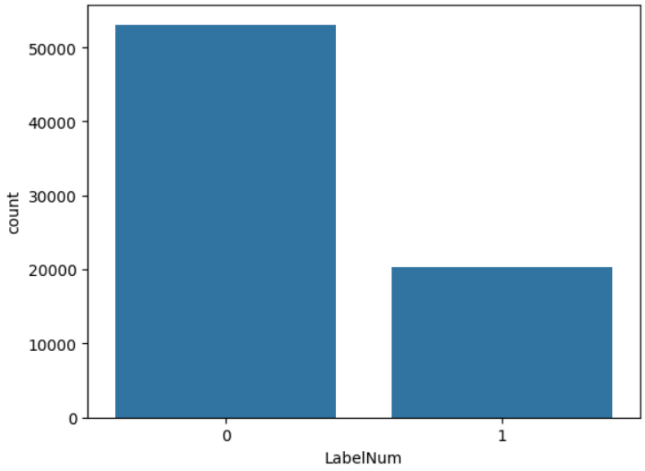

Figure 4: Count of Android (0) vs iOS (1) Questions .

For question classification, [AD23] presents an approach that combines the strengths of Electra (Transformer-based model, GloVe, and LSTM models). It also mentioned that BERT, RoBERTa, and DistilBERT are well-known models used in question classification task.

2.4 Data Analysis and Insights over the Dataset

After analyzing Android VS IOS Dataset, we came up with the following insights regarding the classification of questions either as Android or IOS.

We started by inspecting the distribution of question lengths in order to gain a better understanding of the dataset which will help us in determining the appropriate architecture to be applied on the dataset. For example, if the majority of questions are relatively short, a simpler model might be sufficient. In our dataset, the questions' length are approximately between 40 and 60, thus, we will be using a simple model.

Moreover, we inspected the distribution of class labels. We plotted a count plot that showed that the dataset is imbalanced as it's obvious that android class is exceeding the IOS by approximately 30,000 records.

Following that, we visualized the most frequent words in both classes by visualizing wordcloud by using

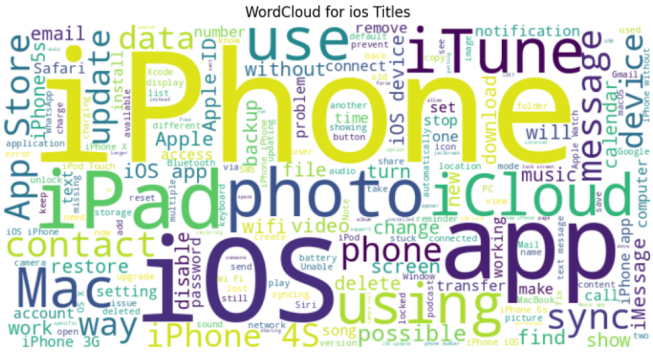

Figure 5: 'WordCloud for iOS Titles'.

Figure 6: 'WordCloud for Android Titles'.

the attribute 'Titles'. For example, questions including the words 'iphone', 'ios', and 'ipad' are most likely to be classified as 'IOS or 1 (Label Number)'. While questions including the words 'android' and 'app' could be classified as 'Android or 0 (Label Number)'. However, we noticed that some questions in the dataset may exhibit overlap between classes, making classification challenging. For example, the word 'app' is considered a frequent word in both classes as visualized in the wordcloud. This insight can guide the inclusion of additional features to differentiate similar questions and avoid ambiguous classification.

3 Methodology

3.1 Problem Statement

A company needs to implement a system that automates the categorization of questions posted into Android or IOS in order to send these questions into the right support team. In addition, the company wants to implement a system that predicts the score of the question to determine how relevant and useful it is.

This project addresses the automation of categorizing questions and predicting their score. A pre-trained BERT model will be used to perform two tasks: Binary Classification - classify the text as either Android or IOS- and Regression Task - predict score of the text. BERT, which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art language model developed by Google. It leverages the transformer architecture to generate rich representations of text by understanding the context of words in a bidirectional manner.

3.2 Binary Classification

This section will explain how we used pretrained BERT model for binary classification task. In order to efficiently explain how the task is executed, it is important to address how the data is prepared, fine tuning of the model and the training process, and lastly evaluating the model.

3.2.1 Data Preparation

To prepare the dataset for modeling, we implemented some preprocessing steps. First of all, it's important to ensure that all columns of the dataset are relevant to the problem. We dropped the columns ID, ViewCount, and Label as they will not be used in solving any of the problems mentioned before.. In addition, we concatenated the columns Title and Body to ensure that the questions are informative. Lastly, we made sure that there are no null values in the dataset used.

Because we are using a predefined BERT model, it is important to prepare the data.Text data is managed and prepared for training a BERT model by handling tokenization, encoding, padding, and truncation of the text samples.

We used CustomDataset class, which is tailored for use in training machine learning models, particularly those based on transformers like BERT. This dataset class includes several methods that encapsulates the data required for training, including the input texts and corresponding labels, along with a tokenizer for tokenization and encoding.
Upon initialization, the class stores the input texts, labels, tokenizer, and a maximum length parameter for tokenized sequences. One of the main methods in this class is "getitem" method which allows indexing to access individual samples from the dataset. Within the "getitem" method, the input text is tokenized and encoded using the provided tokenizer which is BertTokenizer in our case, specifically invoking the "encodeplus" method. "encodeplus" method converts the input text into input IDs and attention mask, which are essential inputs for transformer-based models like BERT. The parameters passed to "encodeplus" ensure that all input sequences have the same length (maxlen) by padding shorter sequences and truncating longer ones, thus facilitating batch processing. Finally, the method returns a dictionary containing the original text, input IDs, attention mask, and label, all appropriately formatted for consumption by the model.

3.2.2 Model Fine-Tuning and Training Process

After preparing the data, we train the model chosen.The model used is BertForSequenceClassification. In this task, we opted for the 'prajjwal1/bert-tiny' model over the 'bert-base-uncased' model due to significantly reduced training times. Initially, the dataset is divided into training and testing sets. training and test splits are then encapsulated within instances of the CustomDataset class, which preprocesses the data, tokenizes and encodes the text inputs, and formats them into suitable representations for training. Subsequently, data loaders (trainloader and testloader) are defined using PyTorch's DataLoader class, facilitating efficient batch-wise loading of the prepared datasets during training and evaluation.

Following data preparation, the training process begins, encompassing a specified number of epochs (3 in this case). Within each epoch, the model undergoes training iterations over batches of data retrieved from the training loader (trainloader). During each iteration, the model is set to training mode (model.train()), and gradients are reset before performing a forward pass through the model. The model's predictions are compared with the ground truth labels, and a loss value is computed using the specified loss function (CrossEntropyLoss). Backpropagation is then applied to compute the gradients of the loss with respect to the model's parameters, followed by an optimization step to update the model's weights. The average training loss across all batches is calculated at the end of each epoch, providing insight into the model's performance and convergence during training.

3.2.3 Evaluation of the Model

After training the model, we evaluate the model on the test dataset. It iterates over the test data batches using the testloader, transferring the input IDs, attention masks, and labels to the specified

device (CPU or GPU). The model generates predictions, and the predicted labels are obtained by taking the class with the highest logit value. The number of correctly predicted labels is accumulated (totalcorrect), and the total number of samples is tracked (totalsamples). Finally, the accuracy is calculated as the ratio of correct predictions to the total number of samples and printed out.

3.3 Regression Task

This section will explain how we used pretrained BERT model for predicting score of a text. In order to efficiently explain how the task is executed, it is important to address how the data is prepared, fine tuning of the model and the training process, and lastly evaluating the model.

3.3.1 Data Preparation

In this task, the data preparation process follows the same methodology as described in Section 3.2.1 of the Binary Classification task.

CustomDataset class was also used in this task. It is initialized with a list of texts, corresponding scores, a tokenizer, and a maximum sequence length. The getitem method retrieves a specific sample by index, converting the text at that index to a string and its associated score to a float. The text is then tokenized and encoded using the provided tokenizer, ensuring that it is padded or truncated to the specified maximum length. The method returns a dictionary containing the input IDs and attention mask tensors, each with the batch dimension removed, along with the score as a tensor of type float. This format prepares the data for efficient loading and use in training or evaluation within a PyTorch model.

3.3.2 Model Fine-Tuning and Training Process

After preparing the data, we train the model chosen. The model used is RegressionBert. In this task, we also opted for the 'prajjwal1/bert-tiny' model over the 'bert-base-uncased' model. Initially, the dataset is divided into training and testing sets. training and test splits are then encapsulated within instances of the CustomDataset class, which preprocesses the data, tokenizes and encodes the text inputs, and formats them into suitable representations for training. Subsequently, data loaders (trainloader and testloader) are defined using PyTorch's DataLoader class, facilitating efficient batch-wise loading of the prepared datasets during training and evaluation.

Following data preparation, a training loop for a regression task over five epochs is executed. Within each epoch, the model (modelregression) is set to training mode, and the total loss is initialized to zero. The training data is iterated over in batches using the trainloader, which contains preprocessed input sequences (inputids) and their corresponding target scores. The model's optimizer gradients are reset, and the model is passed the input sequences to generate predictions. These predictions are compared with the target scores using a specified loss function (MSE), and the loss is calculated. The gradients of the loss with respect to the model's parameters are computed via backpropagation, and the optimizer (AdamW) adjusts the model's parameters to minimize the loss. At the end of each epoch, the total loss is printed, providing insight into the model's training progress and convergence.

3.3.3 Evaluation of the Model

After training the model, we evaluate the model on the test dataset. The test dataset is prepared using the CustomDataset class, which preprocesses the input texts (testtexts) and their corresponding scores (testscores) into suitable representations for evaluation. The test dataset is then loaded into batches using a data loader (testloader) with a batch size of 16 and shuffling disabled. During evaluation, predictions are generated for each batch of input sequences, and these predictions are compared with the target scores using mean squared error (MSE) as the evaluation metric. The MSE is calculated and printed, providing insight into the model's performance on the regression task.

3.4 Discussion

Applying a pre-trained BERT model to regression and binary classification tasks provides a flexible method for handling natural language processing (NLP) issues. We can take advantage of a pre-trained BERT model's rich contextual awareness of language by tailoring it for particular classification or regression tasks. BERT models have been trained on enormous volumes of text data. BERT can discriminate between two classes in binary classification because of its capacity to capture complex semantic links in text. By training the model on a task-specific dataset, one may fine-tune the model's parameters to better fit its learnt representations to the specifics of the target task. Similarly, BERT's rich representations of textual inputs can be used in regression problems, where the objective is to predict continuous values like score.

4 References

References

[AD23] Sanad Aburass and Osama Dorgham. An ensemble approach to question classification: Integrating electra transformer, glove, and lstm. *arXiv preprint arXiv:2308.06828*, 2023.

[AH12] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

[HAA16] Suhaib Kh Hamed and Mohd Juzaiddin Ab Aziz. A question answering system on holy quran translation based on question expansion technique and neural network classification. *J. Comput. Sci.*, 12(3):169–177, 2016.

[KM11] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, 2011.

[MSB13] Nidhi Malik, Aditi Sharan, and Payal Biswas. Domain knowledge enriched framework for restricted domain question answering system. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–7. IEEE, 2013.

[NL15] Mariana Neves and Ulf Leser. Question answering for biology. *Methods*, 74:36–46, 2015.

[RSJ10] Santosh Kumar Ray, Shailendra Singh, and Bhagwati P Joshi. A semantic approach for question classification using wordnet and wikipedia. *Pattern recognition letters*, 31(13):1935–1943, 2010.

[SGW+23] Lei Su, Jiazhi Guo, Liping Wu, Han Deng, et al. Bamnettl: bidirectional attention memory network with transfer learning for question answering matching. *International Journal of Intelligent Systems*, 2023, 2023.

[SP20] Marco Antonio Calijorne Soares and Fernando Silva Parreiras. A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*, 32(6):635–646, 2020.

[YZK+22] Zhenrui Yue, Huimin Zeng, Ziyi Kou, Lanyu Shang, and Dong Wang. Domain adaptation for question answering via question classification. *arXiv preprint arXiv:2209.04998*, 2022.

[ZLC+23] Wenfeng Zheng, Siyu Lu, Zhuohang Cai, Ruiyang Wang, Lei Wang, and Lirong Yin. Palbert: An improved question answering model. *Computer Modeling in Engineering & Sciences; Tech Science Press: Henderson, NV, USA*, 2023.