Real-time Open-Domain Question Answering from Wikipeadia

Pritam Mishra
Id-35390350

Abstract

The quest for knowledge is deeply human, and so it is not surprising that practically as soon as there were computers we were asking them questions. Using Wikipedia articles as the knowledge source causes the task of question answering(QA) to combine the challenges of both large-scale open-domain QA and of machine comprehension of text. This paper considers the problem of answering factoid questions in an open-domain setting using Wikipedia as the unique knowledge source: the answer to any factoid question is a text span in a Wikipedia article. Unlike most QA or reading comprehension models, which are best described as rerankers or extractors since they assume as input relatively small amounts of text (an article, top k sentences or passages, etc.)[2], this system integrates state of the art word embeddings viz. BERT[6], GLOVE-300[5] to not only produce ranking matrices for QA pair, but also utilizes the matrices on WikiQA[4] dataset to train our optimal model and finally choose the best possible answer based on the proprietary ranking algorithm applied here on a large corpus of several wikipedia pages.

1 Introduction

Large-scale QA systems like IBM's DeepQA (Ferrucci et al., 2010)[7] rely on multiple sources to answer: besides Wikipedia, it is also paired with KBs(knowledge Base), dictionaries, and even news articles, books, etc. As a result, such systems heavily rely on information redundancy among the sources to answer correctly. Having a single knowledge source forces the model to be very precise while searching for the optimal answer for the question based on linguistic learning algorithms as the evidence might appear only once. As NLP researchers became increasingly interested in QA, they placed greater emphasis on the later stages of the pipeline to emphasize various aspects of linguistic analysis[2]. This challenge thus encourages research in the ability of a machine to read, a key motivation for the machine comprehension subfield and the creation of datasets such as SQuAD (Rajpurkar et al., 2016)[8], CNN/Daily Mail (Hermann et al., 2015)[9] and CBT (Hill et al., 2016).

However, those machine comprehension resources typically assume that a short piece of relevant text is already identified and given to the model, which is not realistic for building an open-domain QA system. In sharp contrast, methods that use KBs or information retrieval over documents have to employ search as an integral part of the solution. This project is focused on simultaneously maintaining the challenge of machine comprehension, which requires the deep understanding of text, while keeping the realistic constraint of searching over a large open resource. BERT (Devlin et al., 2018)[6], the latest refinement of a series of neural models that make heavy use of pre-training (Peters et al., 2018; Radford et al., 2018), has led to impressive gains in many natural language processing tasks; has been implemented in this project for both feature extraction and ranking metric. This project has extensively bulit and validated our QA model based on WikiQA[4](Yang et al., 2015) data-set and finally the predicted answers by the model (corpus of several pages returned on search query from Wikipedia Server) have been ranked(proprietary ranking algorithm) to provide the user top 5 optimal answers for the query.

2 Related Work

While the origins of question-answering date back to the 1960s,the modern formulation can be traced to the Text Retrieval Conferences (TRECs) in the late 1990s (Voorhees and Tice, 1999)[11]. With roots in information retrieval, it was generally envi-

sioned that a QA system would comprise pipeline stages that select increasingly finer-grained segments of text (Tellex et al., 2003): document retrieval to identify relevant documents from a large corpus, followed by passage ranking to identify text segments that contain answers,and finally answer extraction to identify the answer spans[2]. Most popular QA benchmark datasets today—for example, TrecQA (Yao et al., 2013), WikiQA[4] (Yang et al., 2015), and MSMARCO (Bajaj et al., 2016)[12]— are best characterized as answer selection tasks. That is, the system is given the question as well as a candidate list of sentences to choose from. Of course, those candidates have to come from somewhere, but their source lies outside the problem formulation.

Similarly, reading comprehension datasets such as SQuAD (Rajpurkar et al., 2016)[8] eschew retrieval entirely, since there is only a single document from which to extract answers. Likewise, our model has been trained and validated on WikiQA[4] (Yang et al., 2015) data-set before applying on a document retrieval task followed by answer prediction and ranking. Since it is impractical to apply inference exhaustively to all documents in a corpus with current models (mostly based on neural networks), this formulation necessarily requires some type of term-based retrieval technique to restrict the input text under consideration and hence an architecture quite like the pipeline systems from over a decade ago. Recently, there has been a resurgence of interest in this task, the most notable of which is Dr.QA (Chen et al., 2017)[1]. In this project rather than using bigram hashing and TFIDF matching document retrieval, smart query reformulation[3] has been applied for consistant document retrival. Other recent papers have examined the role of retrieval in this end-to-end formulation(Wangetal., 2017;Kratzwal d and Feuerriegel,2018;Lee et al., 2018), some of which have, in essence, rediscovered ideas from the late 1990s and early 2000s.

3 Methodology

In this section, the methodology to answer open-domain questions from wikipedia is discussed as follows,

3.1 Architecture

The architecture of this open domain question answering system can be explained with the help figure 1. Initially, the system applies some pre-processing to the texts question and answer pair as shown in figure 1. The question is reformulated to a sentence by removing the question word and question mark '?'. After this, the sentences (question and answer pair) were vectorized either by word embeddings (GLOVE-300[5], BERT[6]) for sentence vectorization or normalized word vectorization by TFIDF after lemmatization and removing Stop-words. Finally these vectors were applied to compute several distance matrices which would be discussed in the following section in detail. These computed distance metrics among the question and answer pair along with additional features like common NER(Named Entity Recognition), Common Noun chunks, Dependency parsing are trained and validated through a GridSearch with several supervised learning algorithms with tuned parameters with scaling in order to obtain the optimal model. The model finally tested on the WikiQA[4] data-set for benchmarking. The second phase of the system includes user provides a query to this system in real-time. The system receives the query, reformulates it and retrieves relevant wikipedia pages from the wikipedia server. The consolidated corpus of the pages are then pre-processed and tokenised for feature extraction with distance matrices in the same way compared to the text from WikiQA[4] data-set. Finally, the features were tested with the already trained model and the predictions were ranked by the proprietary algorithm of this project.

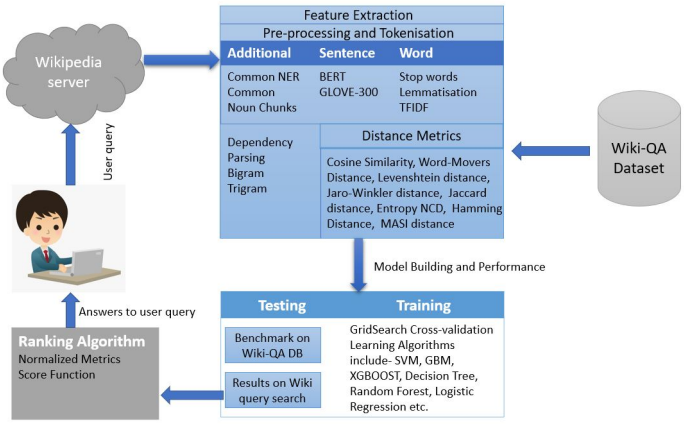

Figure 1: System Architecture

3.2 Pre-Processing

In this project, separate pre-processing has been applied to data from WikiQA[4] data-set compared to data from wikipedia server pages. For the data from

WikiQA[4] data-set smart query reformulation has been developed to transform the question into a sentence. This includes removing any question words or question mark ('?') so that it could obtain meaningful results on the distance based or similarity measures between query and answer. The query formulation has been done for both WikiQA[4] data-set and user query. However, on wikipedia pages that returned from user query, further preprocessing has been applied to remove '@','#' symbols along with any url or unwanted text. The text beyond reference section of the wikipedia pages have also been removed in this process.

3.3 Tokenization and encoding

This project has focused on both sentence based tokenizers and word based tokenizers to make use of unique features which would be discussed in the next section. The sentence based tokens were further encoded by pre-trained word embedding models (GLOVE-300[5], BERT[6]). The power of contextual embeddings allow question answering models based on BERT[6] contextual embeddings and the transformer architecture to achieve even higher accuracy compared to ELMO[10] or bi-LSTM based model (Chen et al. (2017)). The word based tokens were normalized using TFIDF vectorizer to produce unbiased score in distance metrics discussed later.

3.4 Feature Extraction

The extracted features for this project can be categorised as follows,

3.4.1 Common Named Entity Recognition

Named entity recognition is the task of identifying and categorizing entities to extract information from the text. This could be helpful to obtain high-level overview of a large quantity of text. NER detects the name of the entity and categorize the entity for better inference. For example, in the following example we can see Apple, UK and 1 bilion are the entities and ORG is the category for organisation. Common NER has been obtained between question answer pair for modeling our classifier.

3.4.2 Common Noun Chunks

Noun chunks are the base noun phrases with description of the noun in the sentence. It works in a similar way to extract information from a large text like NER.

3.4.3 Dependency Parsing

Dependency Parsing is the task of recognizing a sentence and assigning a syntactic structure to it. In this project, dependency parsing has been applied to find common noun and verb chunks in question-sentence pair. The parsing tree for a question like "how long was richard nixon a president ?" would produce the following,

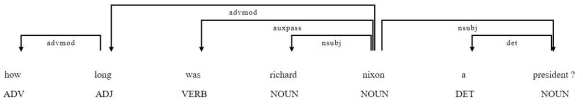

Figure 2: Dependency Parsing

3.4.4 Distance Based Metrics

These metrics were responsible to determine how similar the answer is compared to the question based on the distance between them. These metrics can be categorized as follows,

- **Edit Distance Based:** These algorithms compute the number of operations needed to transform one string to another. More the number of operations mean less similarity between the two strings. Hamming distance, Levenshtein distance, Jaro-Winkler Distance from edit based algorithms have been applied to obtain similarity measure in this project.
- **Token-based:** These algorithms take set of tokens as input compared to strings. The idea is to find common tokens which means more tokens implies more similar they are. A string can be split into tokens or n-gram characters where tokens of different length have equal importance. Jaccard Distance, Sorensen-Dice Distance, Cosine Distance from token based algorithm have been applied to obtain similarity measure in this project.
- **Sequence-based:** The similarity for sequence based algorithm is measured as a factor of common sub-strings between the two strings. The algorithms find the longest sequence present in both strings. The more of these sequences found, higher is the similarity score. Ratcliff-Obershelp Distance has been applied to obtain similarity measure in this project.

MASI distance and Normalized Compression Entropy were also applied in this project for extracting new features from the tokens.

3.5 Data

(Yang et al., 2015)[4] is a sentence-level QA dataset, containing 1.9k/0.3k train/dev answerable examples. Each example consists of a real user's Bing query and a snippet of a Wikipedia article retrieved by Bing, containing 18.6 sentences on average.The task is to classify whether each sentence provides the answer to the query. There are in total 20349 rows out of which 1039 are positive class and the rest are negative class. There is huge class imbalance since the idea is to answer multiple choice questions. However, in this project total data has been undersampled to 2000 rows containing 1000 rows of positive and negative samples to avoid any performance bias. The column names of the data-set includes QuestionID, Question, DocumentID, DocumentTitle, SentenceID, Sentence, Label. The data-set can be found from the Microsoft website. Mostpopular QA benchmark datasets today—for example, TrecQA (Yao et al., 2013), WikiQA[4] (Yang et al., 2015), and MSMARCO (Bajaj et al., 2016)[12]—are best characterized as answer selection tasks[2].

3.6 Training and Classification

Once all the features have been extracted from the WikiQA[4] data-set, the feature-set along with the trained labels were applied on a GridSearch algorithm with stratified 5 fold cross-validation. The supervised algorithms that were applied in the GridSearch model pipeline includes Logistic Regression, Random Forest, Support Vector Machines, Decision Tree, XGBOOST, GBM etc. The data has also been standardised so that it scales well with all other features without bias. Once the data-set has been trained and validated on 80% of the data, it is tested on the test set for benchmark purposes. Further, the model has been tested again when feature set were constructed on wikipedia pages from real-time query of the user. As per the architecture the predicted answers were sent to ranking algorithm for top 5 answers for the user query.

3.7 Ranking Algorithm

In order to rank or provide scores to predicted answers from the model, an algorithm has been developed. According to this proprietary algorithm, all the distance metric score have been scaled proportionally using a MinMax Scaler. The scaled or normalized scores of these features are then subtracted with unscaled value of common NER and common Noun chunks. The resultant score has been ranked accordingly and top 5 answers in ascending order were returned to the user.

4 Experimental Results

Training and validation stage of this open-domain question answering system is consequential in predicting the answers for the questions in real-time. Therefore, several experiments were performed to obtain the optimal model from the training stage. In this project, GridSearch with stratified 5 fold cross validation have applied to train and validate our data-set with maximum efficacy. Parameter Tuning is a major step here and can sometimes be very challenging as it becomes computationally expensive to train the model for several parameters. Therfore, in these experiments it has been observed, the top 3 algorithms which is consistently obtaining optimal performance without using random state. Once, it has been determined, parameter tuning for those algorithms have been done to obtain the model with best parameters. It has been observed after several experiments, Logistic Regression, SVM and XGBOOST were performing consistently to obtain best overall performance. The performance of the final model can be demonstrated with the help of the following figure,

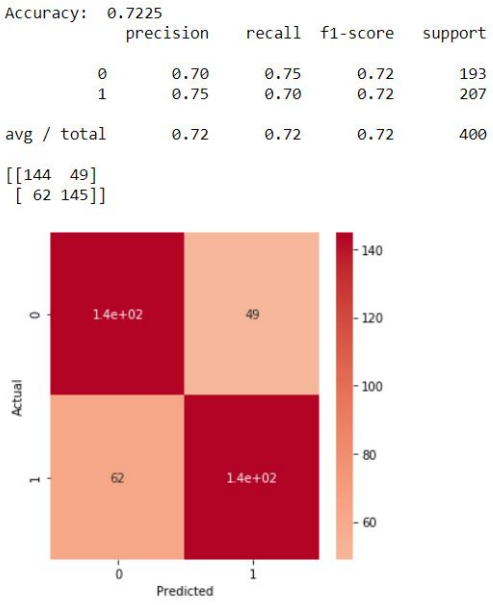

Figure 3: Results: GridSearchcv

As we can see from Figure 3, the average precision, recall and f1-score are all 72% obtained on our final model for this data-set. There is negligible imbalance in the test data so accuracy, precision, recall and f1-score should not be misleading for these experiments. In order to be

full sure, let us look at the ROC curve for this result and look at the area under the curve(AUC).

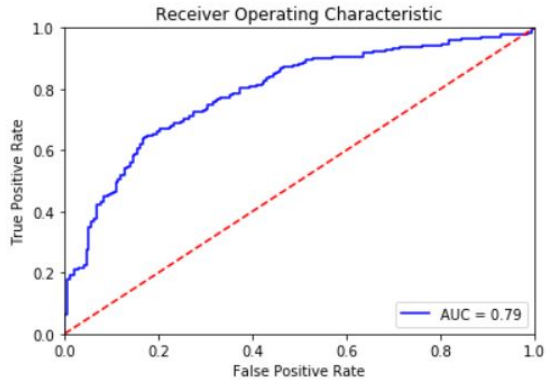

Figure 4: ROC curve

The area under the ROC curve (AUROC) of a test can be used as a criterion to measure the test's discriminative ability, that is it represents degree or measure of separability. From Figure 4, we can conclusively say that there is 79% chance that our final model will be able to distinguish between positive class and negative class.

5 Findings

Based on the research and the methodologies implemented to address the research problem, it seems real-time open-domain question answering from wikipedia is a very challenging problem. Compared to bi-LSTM based model on GLOVE[5] embedding implemented in chen et at. 2017, Google's BERT (Devlin et al.,2018)[6] with 12 attention layers provides state-of-the-art sentence embedding for contextual meaning of sentences with unparalleled performance which was first introduced in ELMO[10] to extract contextual meanings from text. In order to fine tune BERT[6] for generating word embedding can be found as beyond the scope of this project. However, the rigorous methodology implemented in this project to integrate real-time question answering with fine-tuning model from WikiQA[4] dataset constructing features with embedding distance metrics from BERT[6], GLOVE-300[5] based raised the horizon of this project. The state-of-the-art linguistic features like NER, Noun chunks, Dependency parsing has improved the model precision and performance significantly. This approach can be considered a novel approach compared to the previous approaches of only ranking the answers based on question answer pair metrics. This model can be further fine-tuned and trained on sQuad[8] data-set and further tested on WikiQA[4] for performance measure(Tey et al.).

6 Conclusion and Future Work

Although, the classifier applied in this project has achieved 72% accuracy, it becomes computationally expensive to tune the parameters of these classifiers. Based on the previous work on open-domain question-answering this novel approach has managed to achieve very impressive performance in terms of average precision, recall and f1-score. The idea can be further expanded if all the layers of BERT[6] can be studied in detail to extract sentence embedding from not only the last or second last layer rather combination of several attention layers. Model performance can also be further improved with ngram-based tokenisation for similarity based distance metrics. Entity-linking can also be studied in detail which in turn can become a consequential feature along with dependency parsing and NER. The biggest improvement in performance can be produced if NER category for the answer can be predicted based on the question type "when" or "where" will have different NER in the answer. This methodology could filter our wikipedia text sentences which could be used for model prediction. One major future work on this project could be a sophisticated document retrieval system like Dr.QA(Chen et al. 2017); a module using bigram hashing and TF-IDF matching designed to, given a question, efficiently return a subset of relevant articles.

References

[1] Chen, D., Fisch, A., Weston, J., and Bordes,A.(2017). "Reading wikipedia to answer open-domain questions. In ACL 2017."

[2] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin. "End-to-End Open-Domain Question Answering with BERTserini"

[3] Daniel Jurafsky , James H. Martin . "Speech and Language Processing -An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition "

[4] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. "WikiQA: A challenge dataset for open-domain question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing".

[5] Jeffrey Pennington, Richard Socher, Christopher D. Manning" GloVe: Global Vectors forWord Representation"

[6] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pretraining of deep bidirectional transformers for language understanding.“ In NAACL HLT 2019,

[7] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. “Building Watson: An overview of the DeepQA project.“

[8] PranavRajpurkar,JianZhang,KonstantinLopyrev,and PercyLiang.2016.“ SQuAD:100,000+questionsfor machinecomprehensionoftext. InEmpiricalMethods in Natural Language Processing“

[9] Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. “A thorough examination of the CNN/Daily Mail reading comprehension task. In Association for Computational Linguistics (ACL)“

[10] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. “Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies“

[11] Ellen M. Voorhees and Dawn M. Tice. 1999. “The TREC-8 Question Answering Track evaluation.“ In Proceedings of the Eighth Text REtrieval Conference

[12] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, MirRosenberg,XiaSong,AlinaStoica,SaurabhTiwary, and Tong Wang. 2016.“ MS MARCO: A human generated MAchine Reading COmprehension datase“