Inter IIT Tech Meet 11.0

DevRev's Expert Answers in a Flash: Improving Domain-Specific QA

End - Term Report

Submitted by :
Team 43

Abstract

This documentation of the final evaluation summarises the Domain-Specific Question-Answering methodologies. In addition, we highlight our team's implemented pipeline and its performance.

Contents

1. Introduction

With technological advancements and a rapid increase in customer demand, developers today face a massive volume of customer queries. There is a widespread effort to reduce the volume of these queries handled manually by letting the system handle them initially and responding automatically with some suggestions. In our case, we create an end-to-end pipeline that, given a user query, returns an answer from the knowledge base articles in order to reduce human effort. This report provides a detailed analysis of our proposed solution and the methodologies used to solve the problem.

2. Implementation

Domain-based question-answering is a two-step process which involves - identifying relevant context and answering the given query from the selected context. For retrieving the relevant context from the knowledge base, the knowledge base is reduced to sentence banks for each theme, these sentences are tokenized and further converted into vector embeddings. Relevant context is identified on the basis of the semantic similarity of the sentence embeddings with the query embeddings. Once the context is identified it is piped into the question answer model along with the query. Backed by extensive research and experimentation, we have incorporated the following methodologies for the tasks.

2.A. Context Retrieval

The first stage of the process involves retrieving the relevant information from the knowledge base with the help of the user's given query and then directing it to the question-answering model. The following are the details of each step involved in this stage.

2.A.I. Generating Embedding

The raw context from each theme is first broken down into sentences, tokenized and converted into vectorized representations using a pre-trained embedding model. Selecting the right embedding model is crucial for giving right weightages to the content of the sentences from the knowledge bank. On experimenting with various embedding models inclusive of DistilRoberta[10], Google's Universal Sentence Encoder(GUSE) [2] for QA, MPNet[13], SimCSE [6], MiniLM[15], Infersent, DPR Reader [9], we found the following to be the best performing in our case.

Sentence Encoder	Dim.	Relevant Para@10[1]	Answer in top 10 sentences[2]
MPNet	768	**96.95** %	**93.21** %
DistilRoberta	768	96.63 %	91.41 %
GUSE for QA	512	95.33 %	90.5 %
SimCSE	768	86.43 %	80.78 %

Table 1: Experimentation Results

MPnet and **DistilRoberta** being used are pre-trained multilingual transformer-based models which efficiently create 768-dimensional *contextual embeddings*. These embeddings capture cross-correlation between different words in the context. This provides higher accuracy but at the cost of increased average embedding generation and search time by a few milliseconds.

Google Universal Sentence Encoder (GUSE) is a pre-trained *Deep Averaging Network* that creates *contextual embeddings* of 512 dimensions. Although the average combined embedding generation and searching time is approximately 10ms, it is less accurate than its transformer-based counterparts.

SimCSE sentence encoder utilises the idea of contrastive learning. It takes *entailment pairs* as hard positive samples and *contradiction pairs* as hard negatives. Therefore, it brings answers containing sentences closer to the query. We used a pre-trained model which generates 768-dimensional embeddings and induces latency similar to transformer-based models.

2.A.II. Searching Algorithm

Once the queries and sentences are converted to vector embeddings, a robust semantic

[1] Percentage of answerable queries for which the relevant paragraph containing the answer has a sentence in the top 10 nearest sentences retrieved

[2] Percentage of answerable queries for which the answer to that query is present in the top 10 nearest sentences retrieved

similarity-based searching algorithm is required to identify the correct context.

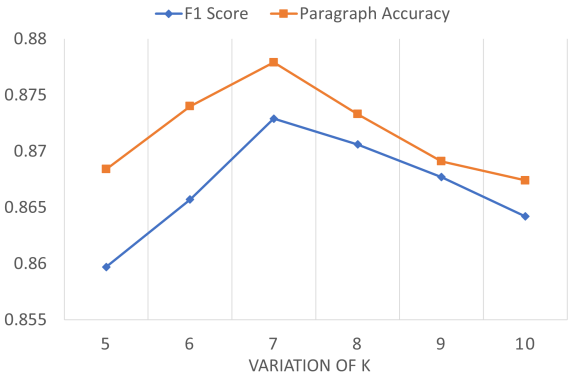
Algorithm	Relevant Para@10	Answer in top 10 sentences
FAISS	**96.95** %	**93.21**
Elastic Search + BM25	90.32 %	88.43 %
ANNOY	73.63 %	69.13 %

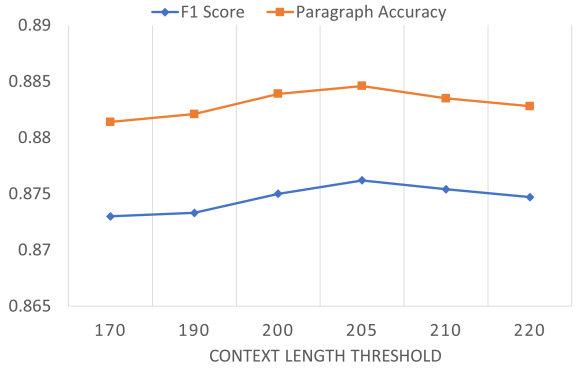
Table 2: Searching Experimentation Results

For experimentation, we used a similarity search algorithm on MPNet-generated embeddings. In our case, **FAISS**, based on comparing L2 distances of the embeddings, gives the best results. **ANNOY**, which is an approximate nearest neighbour-based method, malforms due to its inefficiency in handling high dimensional embeddings. BM25 is a scoring method based on a variation of **TF-IDF** and therefore does not require embeddings. However, it ignores context while clustering and therefore has poor accuracy.

2.A.III. Context Selection

With experimentation, it was observed that the model's performance directly depended on the size of the retrieved context. Therefore, using *Dense Paragraph Retrieval* [9], the based method was useless in our case since it returns the top 10 paragraphs, which act as a noise for the question-answer model, which leads to reduced accuracy and high inference time. We tried further shortlisting content from the retrieved paragraphs—this additional step induced latency in our pipeline. So, we resorted to selecting the top K sentences based on the semantic similarity of the query and the knowledge-bank sentences. Furthermore, we set up the word limit on the retrieved information by introducing *context length threshold*. The figure below shows the F1 score and paragraph accuracy at different values of "K" and "context length threshold".





The figures indicate that a word limit of 205 words on the retrieved context yielded the best results. We also added a *distance_threshold* to limit the context length further. Given *dist[i]* represents the distance of i^{th} sentence from the query, the i^{th} sentence is added to the context if and only if -

$$dist[i] \leq distance\_threshold * dist[1]$$

This ensures that unnecessary sentences are not part of the context, thus improving the chance of QA models capturing the correct answer.

2.A.IV. Fine-Tuning

MPNet generated embeddings coupled with FAISS searching algorithm are giving the best results. Since the MPNet model that was being used initially was pre-trained on a multi-lingual dataset, we further fine-tuned it on a custom english language dataset created from the given train dataset, ensuing an improvement in the results. The fine-tuning methodology was inspired by *Contrastive Learning* technique which was employed in the SimCSE sentence encoder model.

2.A.IV.1. Generating Synthetic Data

For fine-tuning MPNet, a synthetic dataset was generated, consisting of triplets {*query, sentence-1, sentence-2*}. The *query* is a question from the training dataset. *sentence-1* is the sentence containing the answer to the given query, and *sentence-2* is a dissimilar sentence from the theme of the query. The dissimilar sentence is retrieved by taking the top-7 similar sentences and then choosing the first sentence that crosses the threshold of 0.8 cosine distance, which is measured by applying MPNet-base and FAISS similarity. If the

threshold is not crossed, we choose the fifth similar sentence as the dissimilar sentence.

> ***Example:***
> ***Query:*** *Where were rural children employed?*
> ***Sentence:*** *Rural areas similarly saw families deploying their children in agriculture.*
> ***Dissimilar Sentence:*** *Vast majority of child labour is found in rural settings and informal urban economy; children are predominantly employed by their parents, rather than factories.*

2.A.IV.2. Training on the Synthetic Data

The MPNet-base embeddings were trained on the generated dataset using contrastive learning. The idea is to decrease the distance between the query and the sentence containing the answer and increase the distance between the query and the unrelated sentences. This is done by using the TripletLoss as the loss function during training. This approach was inspired by the SimCSE paper [6]. This led to a *2% increase* in the paragraph accuracy on our validation dataset, boosting the overall F1 score by nearly the same margin.

2.B. Question Answering

2.B.I. Models and Performance

Practically speaking, it is not possible to train a large transformer-based network from scratch, which is necessary for this task. Therefore, we utilise several pretrained models and compare their performance.

QA Model	Avg. Inf. Time	Para Accuracy	F1 Score
MiniLM	**126.7 ms**	0.84	0.7995
ALBert-base	741 ms	0.8509	0.7729
RoBerta	567 ms	0.8323	0.7926
RoBerta-distilled	568 ms	0.8663	0.8324
Electra-base	539 ms	0.8519	0.8495
Electra-best	772 ms	**0.8779**	**0.8729**

Table 3: Performance of QA Models

We discovered that MiniLM provides extremely low latency with sufficient accuracy metrics, making it a highly appealing choice for the first time restriction. As soon since it was raised, however, we moved our emphasis to ELECTRA as it provided superior performance in sub 1s.

2.B.II. Fine-Tuning Methods

Given the diverse set of themes, fine-tuning a single model on all the data will not help it capture the granularities present in the themes. Hence, it was favourable to fine-tune the base model on each individual theme. However, given the enormous number of themes in the training corpus, this would be infeasible, both in terms of computation time and space. To circumvent this, we can group the themes into clusters of related themes and fine-tune the model for each topic cluster. During evaluation, the relevant fine-tuned model will be used for answering the questions to achieve the best possible outcome. BERTopic [7] is a commonly used library for implementing topic modelling and was used for this purpose. The main processes involved are outlined in Figure 1.

Creating Embeddings

The first step in the process is the create embeddings of the paragraphs. This is a basic step in any NLP task, and it involves transforming the text into high-dimensional vectors prior to any processing or learning. Google's Universal Sentence Encoder is utilized for this, for the balance it offers between the information in the embedding and the size of the embeddings. First, the paragraphs of a topic are concatenated into an array, and then the resulting embeddings are forwarded to the next block of the model.

Dimensionality Reduction

The next step is to reduce the dimensionality of the data. **U**niform **M**anifold **A**pproximation and **P**rojection for Dimension Reduction [12] is used to eliminate dimensions or features while preserving as much information as possible. This improves the performance of the clustering model.

Clustering Algorithms

A Clustering algorithm is applied on dimension-reduced theme embeddings to group similar themes together in order to extract broad topics. BERTopic is designed to work with a variety of clustering algorithms. The following algorithms were tested:

1) **BIRCH**[16]: **B**alanced **I**terative **R**educing and **C**lustering **H**ierarchies clusters the dataset into summaries prior to clustering the summaries themselves. Other clustering algorithms may also utilise these summaries.

BIRCH	HDBSCAN	K-Means
'The Legend of Zelda: Twilight Princess', 'Computer security', 'Video conferencing', 'Xbox 360', 'Macintosh', 'Dell', 'Nintendo Entertainment System', 'Digimon', 'PlayStation 3', 'IBM', 'Windows 8', 'Super Nintendo Entertainment System'	'Internet service provider', 'Communications in Somalia', 'High-definition television', 'Computer', 'MP3', 'Computer security', 'Video conferencing', 'Xbox 360', 'ASCII', 'Macintosh', 'Dell', 'Nintendo Entertainment System', 'Film speed', 'Data compression', 'Gramophone record', 'PlayStation 3', 'LaserDisc', 'IBM', 'Compact disc', 'Windows 8', 'Super Nintendo Entertainment System'	'BBC Television', 'Internet service provider', 'BeiDou Navigation Satellite System', 'Communications in Somalia', 'Computer security', 'Video conferencing', 'Dell', 'Copyright infringement', 'Intellectual property', 'CBC Television', 'IBM', 'YouTube'
'Architecture', 'Royal Institute of British Architects', 'Westminster Abbey', 'Buckingham Palace', 'Gothic architecture', 'Cubism', 'Neoclassical architecture', 'Georgian architecture', 'Mosaic'	'Bern', 'Chihuahua (state)', 'Hanover', 'Valencia', 'Mexico City', 'Thuringia', 'Switzerland', 'Carnival', 'Galicia (Spain)', 'Strasbourg', 'Paris', 'Palermo', 'Hyderabad', 'Mosaic', 'Brasília'	'Architecture', 'Royal Institute of British Architects', 'Westminster Abbey', 'Buckingham Palace', 'Gothic architecture', 'Cubism', 'Neoclassical architecture', 'Georgian architecture', 'Mosaic'

Table 4: Clustering Algorithms Experimentation Results.

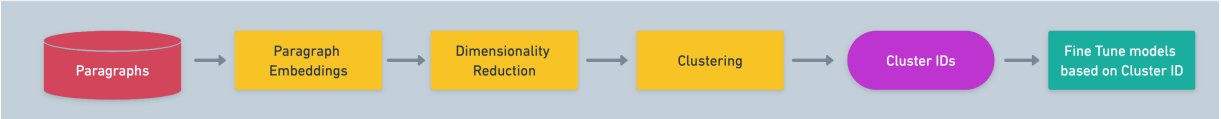


Figure 1: Clustering

Due to the fact that it is a hierarchical model that only requires a single scan of the dataset, it is highly capable of handling massive datasets.

2) **HDBSCAN**[11]: **H**ierarchical **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise) extends DBSCAN by hierarchizing its density-based model. It first transforms the space based on density to identify clusterable islands. It then constructs a minimal spanning tree connecting the data points, which are subsequently evaluated hierarchically, condensed, and extracted as clusters.

3) **K-Means**: K-Means Clustering is an iterative algorithm that attempts to divide the dataset into k distinct clusters. It shuffles the data and selects k centroids at random, and then attempts to cluster the datapoints until the assignment of datapoints to clusters becomes stable. Finally, it assigns each datapoint to its nearest cluster and computes the cluster centroids by averaging the datapoints within each cluster.

Clustering Algorithm	No. of Clusters	Themes per Cluster	Unclustered Themes
BIRCH	37	9.756	0
HDBSCAN	13	16.231	150
MST	15	24.066	0

Table 5: Summary of Clustering Algorithms

After clustering, a tokenizer can be employed to facilitate the creation of a topic representation using c-TF-IDF. This part is not relevant to the given application and hence elaboration is skipped for brevity. BIRCH was finally choosen because of its facility with massive datasets.

Fine Tuning QA Model

For fine tuning, QA is modelled as a span prediction problem; the model is tasked with predicting the start and end tokens given the context and the query. It was decided to fine-tune models on clusters instead of individual themes. The most significant decrease in validation loss while fine tuning was observed while using a linear decay schedule at a learning rate $\frac{1}{20}$ of the pretraining learning rate. Grouped layer-wise learning rate decay **LLRD** - where every group of layers have a constant learning rate instead of decreasing linearly with training steps - gave promising results but at cost of increased training time on CPU. Therefore we resorted to using a linear decay schedule.

Cluster	Base Model	Fine Tuned Model
Wars	0.83436	0.86278
Cities	0.85685	0.89702
US Cities	0.85481	0.89618

Table 6: Effect on fine-tuning QA models

2.B.III. Model Optimization

2.B.III.1. ONNX Runtime

ONNX is an open standard that defines a common set of operators and a common file format to represent deep learning models in various frameworks. By exposing a graph with standardized operators and data types, onnx makes it easy to switch between frameworks. Onnx with a dedicated accelerator like *Onnx Runtime* provides tools to optimize the onnx graph through operator fusion and constant folding to accelerate latency and inference.

Onnx Runtime also provides support for the quantization of models. Quantization is a technique to reduce the computational and memory costs of running inference by representing the weights and activations with low-precision data types like an 8-bit integer (int8) instead of the usual 32-bit floating point (float32). Reducing the number of bits means the resulting model requires less memory storage, consumes less energy (in theory), and operations like matrix multiplication can be performed much faster with integer arithmetic. We performed optimization and quantization of question-answering models using the optimum package of huggingface's transformers library. The results obtained are shown in the Table 7. The quantized models were roughly half in size of the original models and provided 1.5x faster inference in CPU with a minor decrease in F1 score. Although promising, our best QA model, *electra-base-best*, has no support for quantization yet.

2.B.III.2. Transformer Adapters

Storing multiple fine-tuned transformer models for different themes requires a lot of memory since a single model requires about 300-400 MBs to store. A solution to this problem is to keep a single base model and insert layers with adjustable weights for different fine-tuned models. AdapterHub[8]is a framework which allows dynamic 'stitching-in' of these pre-trained layers called adapters, with each adapter being lightweight. This provides results comparable to fine-tuned models with minimal memory use. Currently, it does not support ELECTRA-based models, so it was not used in the final pipeline.

Base Model	Adapter	F1	Inference Time
ROBERTA	roberta-base-pf-sqaudv2	0.71	689 ms
	ukp/roberta-base-pfeiffer	0.781	632ms
	ukp/roberta-base-houlsby	0.750	670ms
BERT	ukp/bert-base-pfeiffer	0.72	640ms
	ukp/bert-base-houlsby	0.746	630ms

Table 8: Effect of Adapters

2.B.III.3. Intel OpenVINO

OpenVINO is an open-source toolkit that enables high performance inference capabilities for Intel CPUs, GPUs, and special DL inference accelerators. It is supplied with a set of tools to optimize and quantize models. On using OpenVINO to quantize **roberta-base-squad-v2**, the model size reduces from 496 mbs to 192 mbs. Inference time also reduces to 550 ms on CPU, with only a slight decrease in the F1 score. However as it does not support Electra currently, we could not implement it in the final pipeline.

2.C. Using Previously Answered Question

We store the query indices and corresponding answers for each answered query. So, when a new query is encountered, semantic similarity search is used in the stored query bank to find the similar query which has been previously answered based on a defined hard threshold on L2 distance. This greatly reduces the average inference time by-passing the major chunk of the pipeline.

Distance Threshold	Recall	Precision	F1 Score
0.1	**1.00**	0.61	0.76
0.2	0.97	0.74	**0.84**
0.3	0.86	0.79	0.82
0.4	0.82	0.80	0.81
0.5	0.74	**0.84**	0.79

Table 9: Scores at different distance thresholds

Table 9 above shows comparison of scores at different distance thresholds. For the experiment, we created a dataset which was a subset of given train dataset. It contains 5000 samples of 500 samples are repeated. Clearly, best results are obtained at the distance threshold of **0.2**.

Question-Answering Model	Number of Parameters	Without Quantization			With Quantization using ONNX		
		Model Size	AIT (in ms)	F1 Score	Model Size	AIT (in ms)	F1 Score
MiniLM-Uncased-Base	33 million	127 MB	183 ms	0.80	81 MB	117 ms	0.79
Roberta-Base	124 million	497 MB	802 ms	0.83	285 MB	568 ms	0.81
Electra-Base	106 million	436 MB	787 ms	0.85	233 MB	539 ms	0.80

Table 7: Effect of Quantization

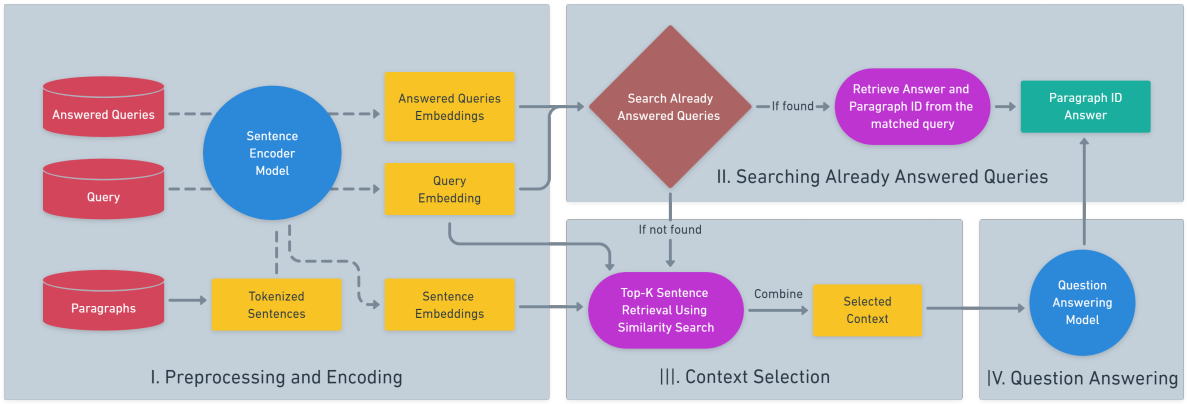


Figure 2: The Final Pipeline

2.D. The Final Pipeline

As shown in the Figure 2, the entire pipeline consists of four sections.

2.D.I. Preprocessing and Encoding

The given train data was first segregated on the basis of theme data. The theme wise data is organised as paragraph bank. It is further converted into sentences and tokenized.These tokens are passed through finetuned-MPnet sentence encoding model and theme-wise sentence embedding bank is maintained. Simultaneously, an unanswered query bank and answered queries bank containing answered queries and corresponding answers is maintained. The Unanswered and answered query banks are sequentially preprocessed and encoded and passed to the second section.

2.D.II. Searching Already Answered Queries

The query embedding is used to search for a similar query in the embeddings bank of already answered queries. If the criterion of distance threshold of 0.2 units is passed, the answer corresponding to the answered query is directly returned else the query embeddings and sentence embeddings bank is piped to section three.

2.D.III. Context Selection

Using the theme-wise sentence embeddings created, the top k nearest sentences to the given query are searched using faiss. The context is then created by combining these retrieved sentences under the constraints -

$k = 10$
$distance\_threshold = 0.2$
$context\_length\_threshold = 205$

2.D.IV. Question Answering

Once the relevant informantion is retrieved, it is piped into the section IV along with the query, the question-answer model *"Electra-base-best"*[4] handles the inputs and returns the answer, *if* present in the retrieved information, *else* returns an empty string.

2.E. Synthetic Data Generation

2.E.I. Question Generation

Research shows that well-generated question-answer pairs boost up the F1 score of QA model almost by 3-4%. Experimenting with various available models, the three best implementations found were: t5-small

based pipeline, synQA-Question-Generator and HayStack QA Pipeline. Table 11 compares the outputs of the most significant models. The synQA model, because of its ability to generate "tougher" questions (using effective negation, paraphrasing and adversarial tricks like providing options for answers) makes it a suitable choice for the task. Comparatively, t5-small and HayStack pipeline usually return QA Pairs which have direct answers (long common subsequences). A few downsides of the synQA model include requirement of answers, long time per question generated and overconfident incorrect questions in case it is provided with garbage input phrases.

2.E.II. Answer Generation

For answer generation, various unsupervised models based on bag of words/tf-idf/rule-based approaches (like Yake, Rake) or noun chunking (Stanford 7-Class NER) fail to generate grammatical phrases suitable for question generation. TextBlob and Supervised models like Kea, do provide more grammatically correct phrases but we get the best results using a combination of Spacy's Pipeline and KeyBERT-KeyphraseVectorizer model. Overall, comparing the outputs with the train_data, we use the ensemble strategy as illustrated in Figure 3 below, which also helps in reducing time required for the task (from 3-4 Q/min to 6-10 Q/min).

***Context**: Beyoncé's first solo recording was a feature on Jay Z's "'03 Bonnie & Clyde" that was released in October 2002, peaking at number four on the U.S. Billboard Hot 100 chart... Refer Question ID 125 in train_data.csv*

QA Model	Top NER Phrases	Top Generated Question-Answer Pairs
Spacy + synQA-Question-Generator	"Best R&B Performance", "Best R&B Song", "Best Contemporary R&B Album", "Best Female R&B Vocal", "october"	Q: When was Beyonce's first solo recording released? | A: October, Q: Which of the following is better: Best Contemporary R&B Album, Best Female R&B Vocal Performance, or Best Rap/Sung Collaboration? | A: Best Female R&B Vocal Performance
TextBlob + synQA-Question-Generator	"beyoncé", "jay z", "bonnie", "clyde", "October", "solo efforts"	Q: What was Michelle Williams and Kelly Rowland known for? | A: solo efforts, Q: Which of the following is African American: Jay Z, Michelle Williams, or Kelly Rowland? | A: jay z
PKE Supervised Model Kea + synQA-Question-Generator	"Love", "beyoncé", "kelli Rowland", "jay z", "June", "crazi"	Q: In addition to Beyonce, who collaborated with Jay Z? | A: kelli Rowland, Q: Beyonce's first solo record was released in what month? | A: June
KeyBert with KeyPhrase-Vectorizers + synQA-Question-Generator	"solo efforts", "awards", "baby boy", "love", "vocals"	Q: Which single, in addition to Crazy in Love, was Beyoncé's first number one single as a solo artist in the US? | A: baby boy, Q: Which of the following is not a single song: "Me, Myself and I", vocals or "Naughty Girl"? | A: vocals
t5-small	-	Q: How many copies has Dangerously in Love sold worldwide? | A: 11 million, Q: Who was Beyonce's first solo artist in the US? | A: Luther Vandross
HayStack QA Pipeline	-	Q: Who was the lead singer of Crazy in LOVE? | A: Jay Z, Luther Vandross, Q: Which song won the Grammys for Best Rap/Sung Collaboration? | A: Crazy in Love

Table 11: Comparison of various Question Answer pairs generating Models

3. Results & Runtime Analysis

The given train data was split into train and validation data. The entire evaluation and results have been obtained by experimenting with the created validation data.

For the experiments, we used various Sentence Encoder models and Question Answering models. As evident from the results (table 12),

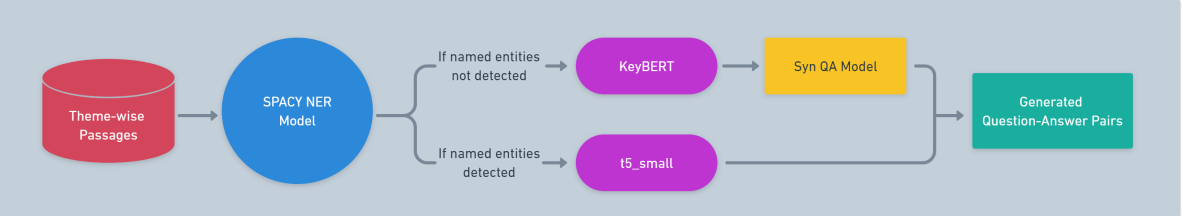


Figure 3: Data Generation Pipeline

Pipeline	Avg Inference Time (ANN + QA = Total)	Answer in context (%)	Para Accuracy	F1 Score
GUSE + MiniLM	7.5 + 126.7 = **134.2** ms	86.44%	0.8014	0.7723
GUSE + MiniLM (fine-tuned)	8.4 + 134.8 = 143.2 ms	86.44%	0.8115	0.7863
MPnet + Roberta-Distilled	71.8 + 568.7 = 640.5 ms	92.08%	0.8724	0.8446
MPnet + Electra-Base	65.4 + 539.4 = 604.8 ms	92.08%	0.8724	0.8446
MPnet + Electra-Best	68.7 + 772.7 = 841.4 ms	92.08%	0.8913	0.8684
MPnet (fine-tuned) + Electra-Best	65.2 + 726.7 = 791.9 ms	94.86%	0.9032	0.8822
MPnet (fine-tuned) + Electra-Best (fine-tuned)	71.5 + 749.6 = 821.1 ms	94.86%	**0.9184**	**0.8975**

Table 12: Performance of Pipeline with various models

Google Sentence Encoder Model, along with MiniLM model, gave the least possible inference time of **134.2 ms** with decent paragraph accuracy and F1 score. We further fine-tuned the MiniLM model to improve the scores. This pipeline is the best considering the earlier constraint of 200 ms average inference time. Presently, in the implemented pipeline, we are using fine-tuned MPnet and Electra-Best to fit into the constraint of 12 hours. Although, the best results are obtained with fine-tuned MPnet and Fine-tuned Electra-Best giving F1 score of **0.8975** and paragraph accuracy of **0.9184**.

4. Conclusion

The report provides a comprehensive research into the task of answering domain-specific questions within an average inference time limit of one second. It includes a comparative study of the performance of various pipeline components, which is supported by experimental results. According to the results, the best performing pipeline in terms of F1 and paragraph accuracy is made up of the fine-tuned Mpnet sentence encoder model and the electra-base-best question answering model. Whereas, the GUSE model and fine-tuned Minilm QA model pipeline outperforms others when it comes to average inference time (∼150ms) with a slight reduction in paragraph accuracy and F1 score. Also, we see a great reduction in average inference time by efficiently handling previously answered queries. The next section presents the future work which can bring significant improvement in this task.

5. Future Work

5.A. Switch Transformers

A significant disadvantage of employing fine-tuned models for each cluster is that without efficiently routing queries between appropriate models, a great deal of processing power and time are wasted, as well as the expense of a large number of parameters associated with each fine-tuned model. Switch Transformers[5] tries to address this by introducing a new routing technique and a novel training method that mitigates the instability problems. It can train a large sparse model with reduced precision costs effectively. The idea is to determine the optimal expert for each token (or query in our example) and route the query only through that subset of the model.

6. Literature Review

We went through various publications in this research area and shortlisted a few that we thought were relevant to our task. The following section contains brief descriptions of the publications.

6.A. Attention is all you Need

The paper[14] introduces a new architecture for machine translation. The paper also argues that the traditional sequence-to-sequence (Seq2Seq) models with recurrent neural networks (RNNs) can be replaced with a simpler architecture based solely on self-attention mechanisms. The resulting model, called the Transformer, achieved state-of-the-art performance on multiple machine translation benchmarks, demonstrating the effectiveness of the attention mechanism. The paper also highlights the benefits of using self-attention over RNNs and convolutional neural networks (CNNs), such as improved parallelizability, reduced computational complexity, and the ability to handle sequences of varying lengths.

6.B. MPNet: Masked and Permuted Pre-training for Language Understanding

The paper discusses a method that involves masking a portion of the input tokens and permuting the remaining tokens to make the model predict the missing and reordered tokens. The paper showed that MPNet outperforms traditional pre-training methods on multiple benchmark datasets for a variety of natural language understanding tasks, including sentiment analysis and question answering. The results suggest that MPNet is effective in capturing the contextual relationships among words and the dependencies between sentence structures and semantic meanings.

6.C. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers

The paper discusses a new method for compressing large pre-trained language models, called MiniLM. The method is based on the idea of distillation, where a smaller model is trained to imitate the outputs of a larger model on a specific task. The paper implements this idea to language models by distilling a small transformer model to imitate the self-attention mechanism of a large pre-trained transformer. The results show that MiniLM can achieve high accuracy on various natural language processing tasks while being much smaller in size compared to the large pre-trained model.

6.D. Google Universal Sentence Encoder

The Universal Sentence Encoder is a pre-trained model for encoding sentences into embedding vectors for transfer learning to other NLP tasks. It is based on two encoding models: one based on the transformer architecture and the other on the deep averaging network (DAN). The models take English strings as input and produce fixed dimensional embedding representations as output. The sentence embeddings can be used to compute sentence level semantic similarity scores and fine-tuned for specific tasks using gradient-based updates. The models are implemented in TensorFlow and made publicly available on TF Hub.

6.E. SimCSE: Simple Contrastive Learning of Sentence Embeddings

The paper presents SimCSE, a simple contrastive learning framework for sentence embeddings. The paper describes two approaches: an unsupervised approach, in which an input sentence is predicted in a contrastive objective using only standard dropout as noise, and a supervised approach that incorporates annotated pairs from natural language inference datasets into the contrastive learning framework. The unsupervised model performs on par with previous supervised counterparts, and the supervised model results in a substantial improvement over prior methods. The authors found that the contrastive learning objective regularizes the pre-trained embeddings and better aligns positive pairs when supervised signals are available.

6.F. AdapterHub: A Framework for Adapting Transformers

AdapterHub is a framework for adapting NLP models based on transformers. The current approach for NLP involves downloading and fine-tuning large pre-trained models, which is expensive and time-consuming. Adapters, which are small learned bottleneck layers, are introduced as an alternative to fine-tuning the entire model. The AdapterHub framework allows dynamic "stitching-in" of pre-trained adapters for different tasks and languages, making it easy and quick to adapt state-of-the-art models such as BERT, RoBERTa, and XLM-R. The framework is built on top of the HuggingFace Transformers library and enables easy sharing and training of adapters, which can be combined with existing models with minimal code edits. AdapterHub provides a website for quick and seamless upload, download, and sharing of pre-trained adapters and is available at AdapterHub.ml.

6.G. A Recurrent BERT-based Model for Question Generation

The paper "A Recurrent BERT-based Model for Question Generation" [3] explores the use of the pre-trained BERT language model for question generation tasks. The paper introduces three neural architectures that use BERT to generate questions from a context text and an answer phase. The first architecture is a straightforward BERT application which shows poor results. The second and third models propose a sequential approach to question generation that takes into account previous decoded results. The models are evaluated on the SQuAD dataset and the best model results in a state-of-the-art performance, advancing the BLEU 4 score from 16.85 to 22.17. The paper concludes that the BERT model can be effectively used for question generation tasks.

6.H. Improving Question Answering Model Robustness with Synthetic Adversarial Data Generation

The paper [1] presents a method for improving the robustness of question answering models by generating synthetic adversarial data. The paper develops a data generation pipeline that selects source passages, identifies candidate answers, generates questions, then filters or relabels them to improve quality. The generated synthetic data is used as part of the training data for a downstream Reading Comprehension model. The paper evaluates the models on the AdversarialQA dataset and MRQA datasets and show that their models are considerably more robust to human-written adversarial examples compared to models trained without synthetic data.

References

[1] Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[3] Ying-Hong Chan and Yao-Chung Fan. A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, Hong Kong, China, November 2019. Association for Computational Linguistics.

[4] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[5] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

[6] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021.

[7] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.

[8] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[9] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[11] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[12] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction.

[13] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding, 2020.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[15] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[16] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, jun 1996.