**CSCE 5290 SECTION\_005 - Natural Language Processing**

**UNT FAQ Response System**

Project Group – ???

Final Report

A green letter on a black background

Description automatically generated

Team:

|  |  |
| --- | --- |
| Name | ID |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Guided by : DR. Jayanth Muthukudage

**Introduction:**

The main objective of this project is to develop a question answering system which help students to navigate through huge information spread across UNT website especially FAQ pages. Our project show cases the start-of-art NLP techniques, with great focus on the implementation of DistilBERT model, as proposed by Sanh et al. (2019). The DistilBert model is a distilled version of BERT which delivers great solutions being smaller, faster and lighter as well as maintaining the high performance required for answering a wide range of questions.

This report aims to talk about the comprehensive methodology used in our FAQ response system. We give a thorough explanation of our methodology, covering everything from the intricacies of data collecting methods—such as web scraping strategies to obtain FAQs—to the optimization of the DistilBERT model in detail. The implementation specifics, difficulties encountered during development, and results obtained are covered in detail in the following sections. Moreover, the integration of a user interface emphasizes our dedication to user-centric design by providing users with good information-seeking experience. This project is proof of our commitment to offering a practical and user-friendly solution to prospective students navigating the abundance of information on the UNT campus websites, in addition to showcasing the practical application of cutting-edge NLP technologies.

**Flow Diagram:**

A diagram of a model

Description automatically generated

**Methodology:**

1. Data Collection:

Data Collection three important steps identifying the URLs, Scarapping the data , writing data scrapped data into Training and Testing Json sets .

1. Identification of URLs : We have 11 websites for scrapping the FAQs.

<https://policy.unt.edu/frequently-asked-questions>

<https://studyabroad.unt.edu/student-faqs>

<https://engineering.unt.edu/students/faq>

<https://informationscience.unt.edu/onsite-institute-faqs>

<https://navigate.unt.edu/faq>

<https://vpaa.unt.edu/advising/get-advised/FAQs>

<https://dining.unt.edu/faq/>

<https://studentaffairs.unt.edu/dean-of-students/about-us/faq>

<https://library.unt.edu/digital-projects-unit/partners/faq/>

<https://tgs.unt.edu/frequently-asked-questions>

<https://clear.unt.edu/supported-technologies/canvas/faq>

1. Scrapping the Data:

We used Beautiful soup library for scrapping the data. One of the most well-known Python libraries for online scraping is called Beautiful Soup; it offers an efficient way to parse HTML and XML documents. Because of its intuitive syntax, it makes it easier to extract information from web pages by making it easier to navigate the document's hierarchical structure and execute focused searches for tags or elements. First, we use ‘requests’ library to make connections and retrieve information from the url. Then we employ Beautiful Soup library for parsing the HTML content of the page. We designed customizable functions to extract the questions and answer from their respective html tags using Beautiful soup.

1. Writing the data into JSON format:

We need the data in a certain format for training, the reason is explained in the model implementation section.

Data Format:

train\_data = [

{

"context": "Mistborn is a series of epic fantasy novels written by American author Brandon Sanderson.",

"qas": [

{

"id": "00001",

"is\_impossible": False,

"question": "Who is the author of the Mistborn series?",

"answers": [

{

"text": "Brandon Sanderson",

"answer\_start": 71,

}

],

}

],

}]

The above is an example data format taken form <https://simpletransformers.ai/docs/qa-data-formats/> to understand the format. We are including this instead of our data because it is smaller compared to our data. The format has,

|  |  |
| --- | --- |
| Context | The paragraph or text from which the question is asked |
| qas | List of questions |
| id | Unique id for the question |
| question | A question |
| is\_impossible | A Boolean value whether the question can answered for the question |
| answers | List of answers |
| text | The answer |
| Answer\_start | Starting index of the answer in the context |

We collected 209 FAQs and divided them into Training and Testing sets in the ratio of 70:30 for training our model.

**Model Implementation:**

We used Simple AI transformer library for the implementation of our Distilbert Question Answering model. This Library supports a great variety of transformers models and has built in support for Text Classification, Question Answering, Language general etc.

from simpletransformers.question\_answering import QuestionAnsweringModel, QuestionAnsweringArgs

From simpletransformers library we import QuestionAnsweringModel where we configure the model type. The QuestionAnsweringModel class support different question answering models like DistilBert, Bert, Roberta, CamemBERT etc.

We used model\_type="distilbert" and model\_name= "distilbert-base-uncased-distilled-squad" to set the required model for our project. Here we used pretrained model on Squad data because pretrained models have several advantages such as

* Pretrained models are already trained on huge datasets which allows them to transfer the knowledge gained during their training to our task.
* Pre-trained models can more effectively generalize to new, unseen data since they have acquired generic features from a variety of data sets. When the data distribution for your task is similar to the pre-training data, this is especially helpful.

**Model Arguments:**

model\_args = QuestionAnsweringArgs(

    reprocess\_input\_data=True,

    overwrite\_output\_dir=True,

    use\_cached\_eval\_features=True,

    output\_dir=f"outputs/{model\_type}",

    best\_model\_dir=f"outputs/{model\_type}/best\_model",

    evaluate\_during\_training=True,

    max\_seq\_length=128,

    num\_train\_epochs=200,

    evaluate\_during\_training\_steps=1000,

    wandb\_project="UNT-FAQ Chat Bot Application",

    wandb\_kwargs={"name": model\_name},

    save\_model\_every\_epoch=False,

    save\_eval\_checkpoints=False,

    n\_best\_size=3,

    train\_batch\_size=128,

    eval\_batch\_size=64,

)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Description** |
| reprocess\_input\_data | True | Even if there is a cached copy of the input data in the cache\_dir, the data will still be reprocessed. |
| overwrite\_output\_dir | True | If True, over the outputs folder after tarining |
| use\_cached\_eval\_features | True | Evaluation during training uses cached features. Setting this to False will cause features to be recomputed at every evaluation step. |
| output | “outputs/{model\_type}” | Stores the model outputs in the mentioned path |
| best\_model\_dir | outputs/best\_model | The path where the best model and its check points are stored |
| evaluate\_during\_training | True | Set to true take the evaluation data when training |
| max\_seq\_length | 128 | Maximum sequence length the model will support. |
| num\_train\_epochs | 100 | The number of epochs the model will be trained for. |
| evaluate\_during\_training\_steps | 1000 | Perform evaluation at every specified number of steps. A checkpoint model and the evaluation results will be saved. |
| wandb\_project | UNT-FAQ Chat Bot Application | W&B project name. All hyperparameter settings, training losses, and assessment metrics will be logged and added to the specified project. |
| wandb\_kwarg | {"name": model\_name} | A dictionary of terms to be sent to the W&B project as arguments. |
| save\_model\_every\_epoch | False | Save a model checkpoint at the end of every epoch. |
| save\_eval\_checkpoints | False | Save a model checkpoint for every evaluation performed. |
| n\_best\_size | 3 | how many of the top-scoring answers the model will consider. |
| train\_batch\_size | 128 | The training batch size. |
| eval\_batch\_size | 64 | The Evaluation batch size. |

Special mention: **Wandb.ai**

The AI developer platform, Weights & Biases (W&B), provides tools for training, optimizing, and utilizing foundation models. Simpletransformer Library enables us to connect wandb website to plot the training loss plot and related information of model while training through an API which is very useful. It is also useful for helpful during fine-tuning the hyperparameters of the model. With the help of Sweep configuration we can configure different sets of hyperparameters for improving the model.

sweep\_config = {

    "method": "bayes",  # grid, random

    "metric": {"name": "train\_loss", "goal": "minimize"},

    "parameters": {

        "num\_train\_epochs": {"values": [50, 75, 100]},

        "learning\_rate": {"min": 5e-5, "max": 4e-4},

        "max\_seq\_length":{"values":[64,128]},

        "train\_batch\_size":{"values":[8, 16, 32]},

        "eval\_batch\_size" :{"values":[8,16,32]}

    },

}

We can see that in the above code we have initialized different sets of parameters, here we pick the parameters in Bayes manner and perform hyper parameter tuning.

def train1():

    # Initialize a new wandb run

    wandb.init()

    # Create a TransformerModel

    model = QuestionAnsweringModel(

        model\_type,model\_name, args=model\_args,sweep\_config=wandb.config,

    )

    # Train the model

    model.train\_model(train\_data, eval\_data=test)

    # Evaluate the model

    model.eval\_model(test)

    # Sync wandb

    wandb.join()

wandb.agent(sweep\_id, train1)

The above function trains the model in with a different set of parameters and give out the best configuration.

Our best obtained parameters, also present in config.json in the project folder:

A close-up of a text

Description automatically generated

**Model performance:**

With limitations on availability, we were unbale achieve a good performance from the model. Also, the process of data collection maybe has hindered the performance of model. We can view different charts in the wandai about the performance of the model.

A graph with numbers and lines

Description automatically generated

The Training\_loss is approaching to zero which means that the model loss is good during the training.

A graph with blue lines

Description automatically generated

Similar means how closely the generated answers are closely related to original answer. Around 61-62 answers are predicted to be similar during sweeping for every iteration. This tells that the model should be trained more with iterations and with sufficiently clean data.

A graph with numbers and lines

Description automatically generated with medium confidence

The above chart shows that there are no exact matches because of long length of answers and less training, even a difference in spacing is considered incorrect predictions.

**Results:**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Future Work:

* Emphasize on the data collection, extracting meaning full context, include data preprocessing.
* Training the model with higher GPUs with different sets of parameters to improve the performance of the model.

**Conclusion:**

To sum up, our project effectively accomplished its main goal of creating a productive question-answering system by utilizing cutting-edge natural language processing techniques, with a specific emphasis on the DistilBERT model. Through the implementation of a thorough methodology that includes web scraping for data collection, careful optimization of the DistilBERT model, and the integration of an intuitive user interface, we have developed a system that is capable of meeting the information-seeking requirements of students who are navigating the extensive content of UNT's website, particularly the FAQ pages. The implementation specifics, obstacles faced, and outcomes attained . This project highlights our dedication to offering a workable and user-centric solution for improving the accessibility of information on the UNT campus websites, in addition to showcasing the real-world use of cutting-edge NLP technologies.

**Note to mention:**

If we are using just pretrained model, trained on Squad data we get better results this shows how important data and training the model are.

**References:**

1. <https://simpletransformers.ai/docs/qa-model/>
2. <https://arxiv.org/abs/1910.01108>
3. <https://docs.wandb.ai/quickstart>