**Open-Domain Question Answering**

Presenters:

Miss. Sowgna\_2010030344

Miss. E.Pravallika\_2010030046

Miss. Tahseen Begum\_2010030168

Miss. Keerthana Pulugam\_2010030445

Guide: Chanda Raj Kumar Rao

**Introduction:**

* Open-ended Open Questionnaire (OpenQA) is an important function in Natural Language Processing (NLP), which aims to answer a question in a natural language based on large informal texts.
* Natural language processing (NLP) refers back to the branch of laptop science—and more specifically, the branch of synthetic intelligence or AI—concerned with giving computers the capability to understand textual content and spoken phrases in lots the equal way human beings can.
* Sub – domain 🡪Open-domain Question Answering

**Problem Statement:**

* Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language. To accomplish this, we will use a Python library and TensorFlow wrapper that makes deep learning and AI.

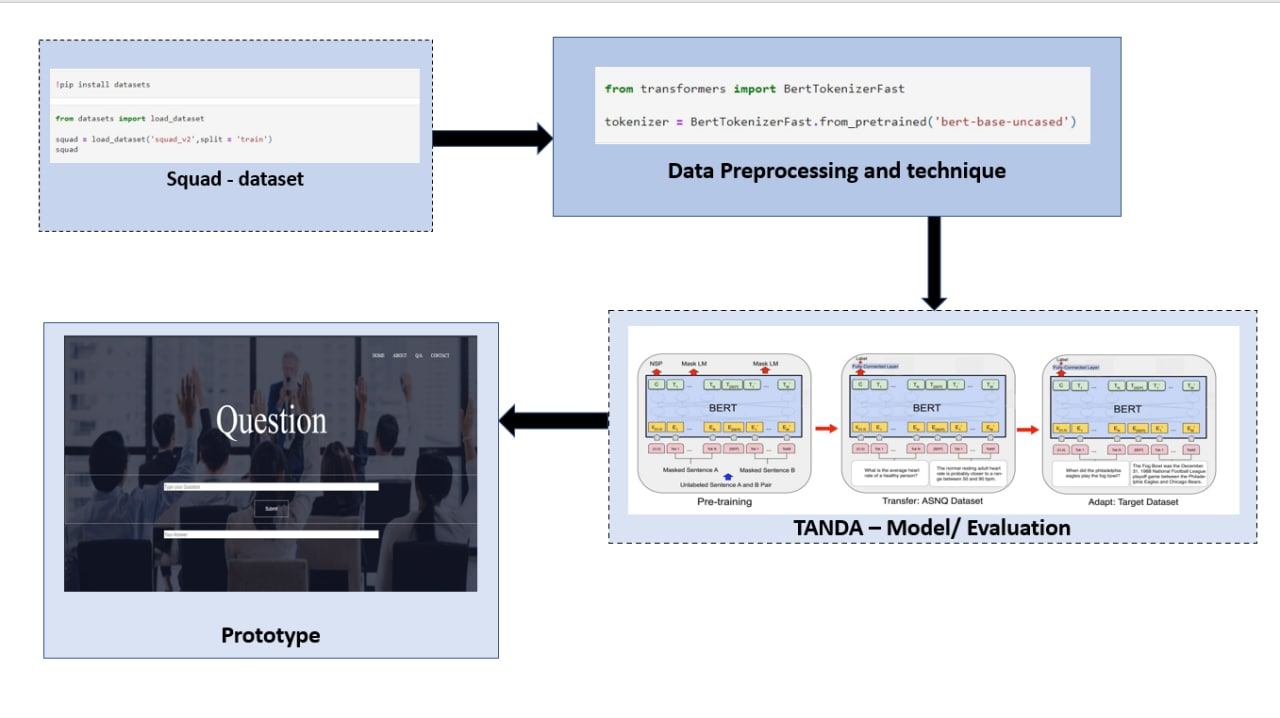
**Motivation:**

* Aims to answer the question in the form of a natural language based on large informal texts.

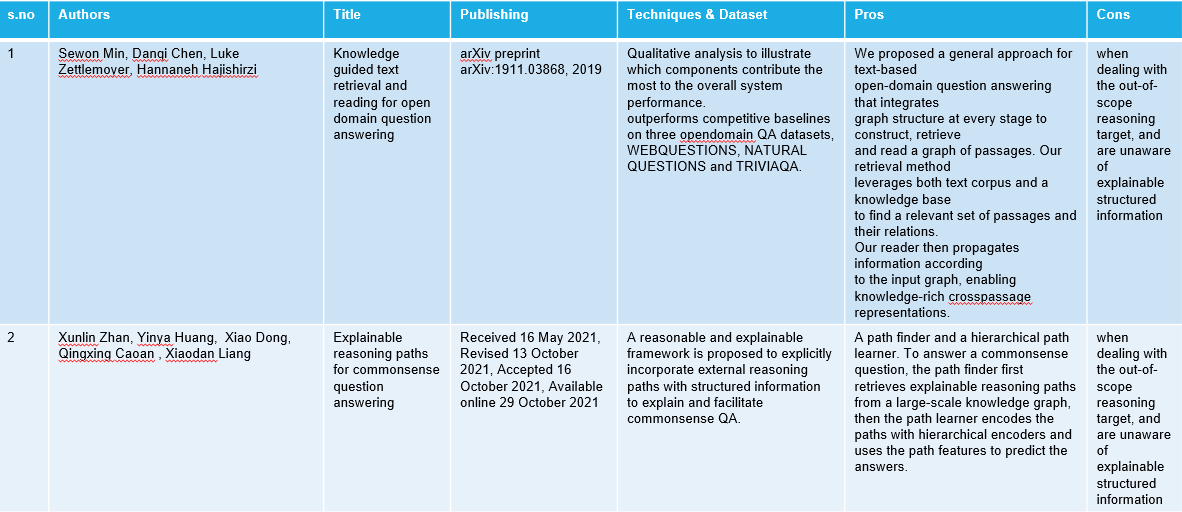
**Objectives:**

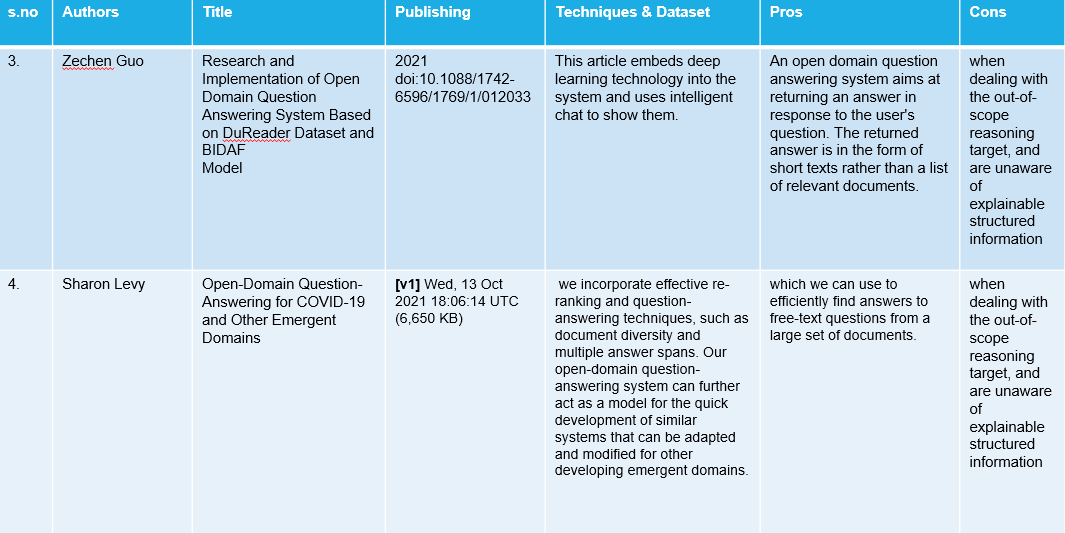
* Objective of the question answering system (QA) is to generate brief answers to the summary questions asked in natural language.
* This kind of information retrieval is required with the growth of digital information.
* Previously QAS were developed for a specific domain and had limited efficiency.
* Present QAS Target on types of questions commonly asked by users, characteristics of data source, and correct answer generated. We aim to build a web-scale QA system
* Most QA systems before answer extraction do question classification for predicting entity type of answer of the question.

**Flowchart**

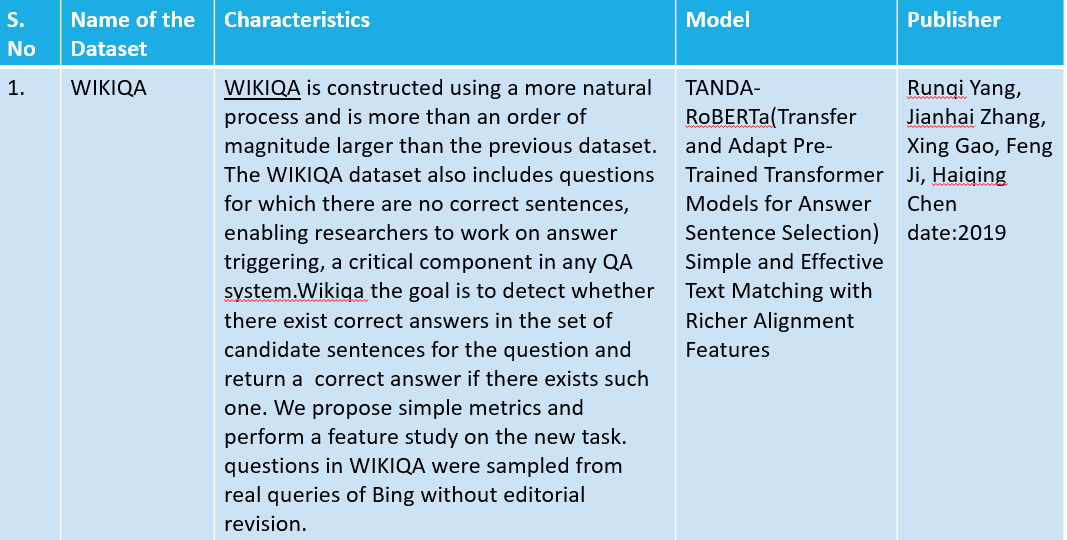
****

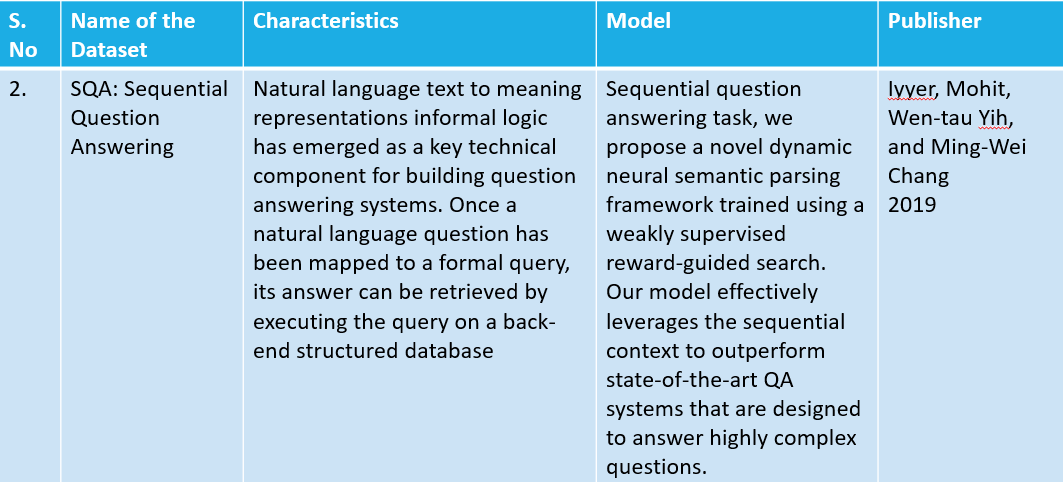
**Literature Survey**

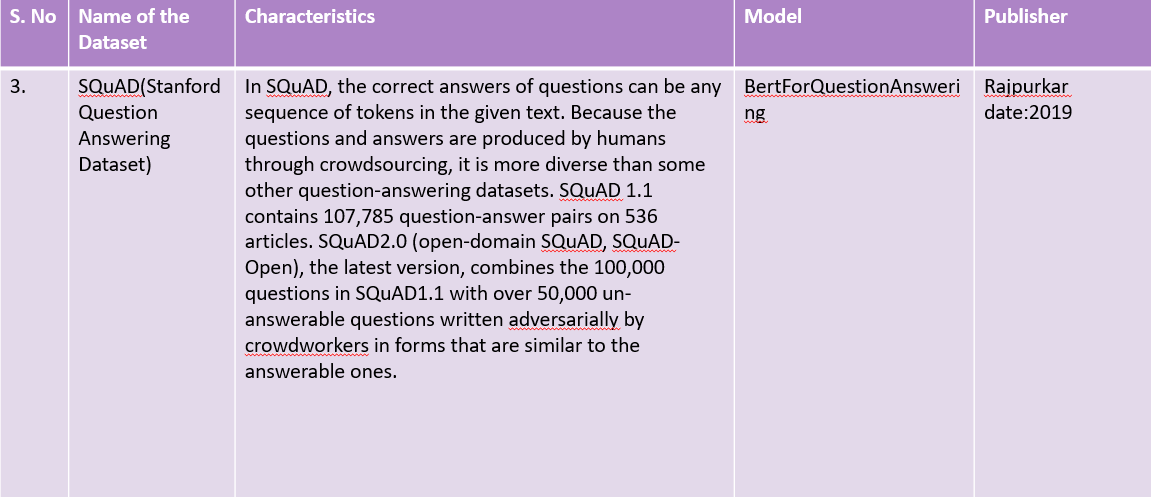
****

****

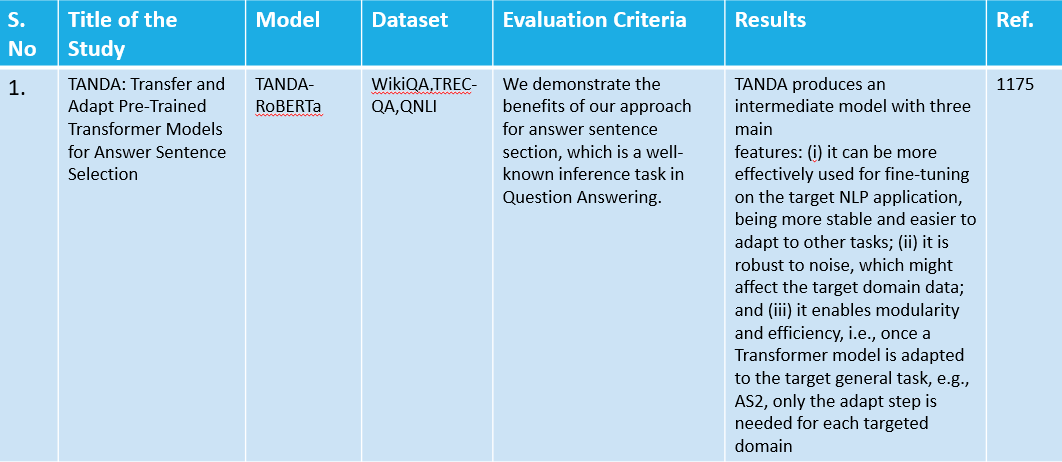
**Dataset:**

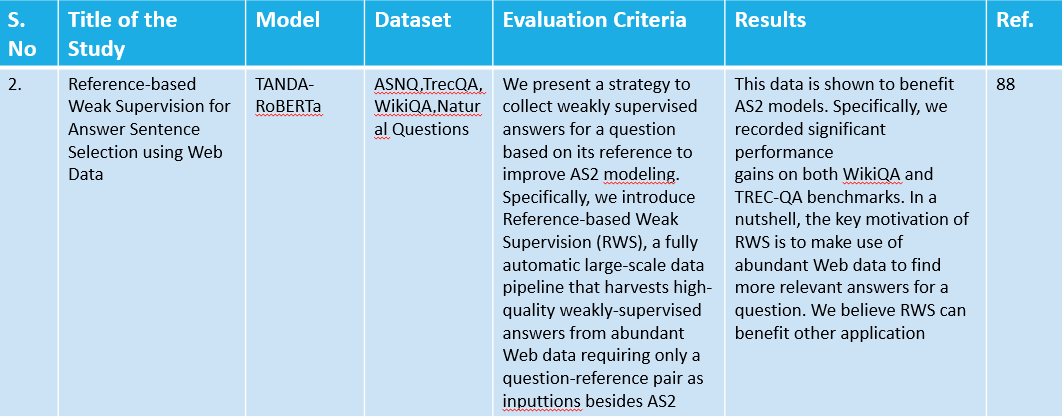
****

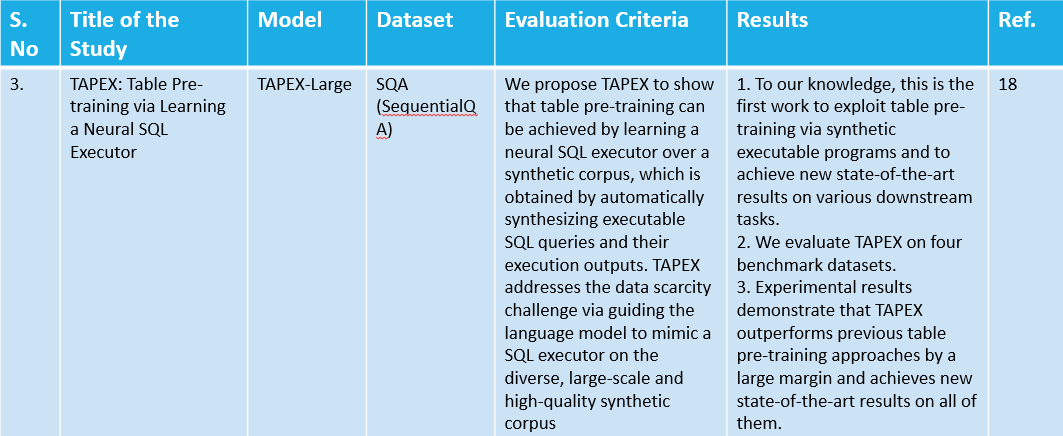
****

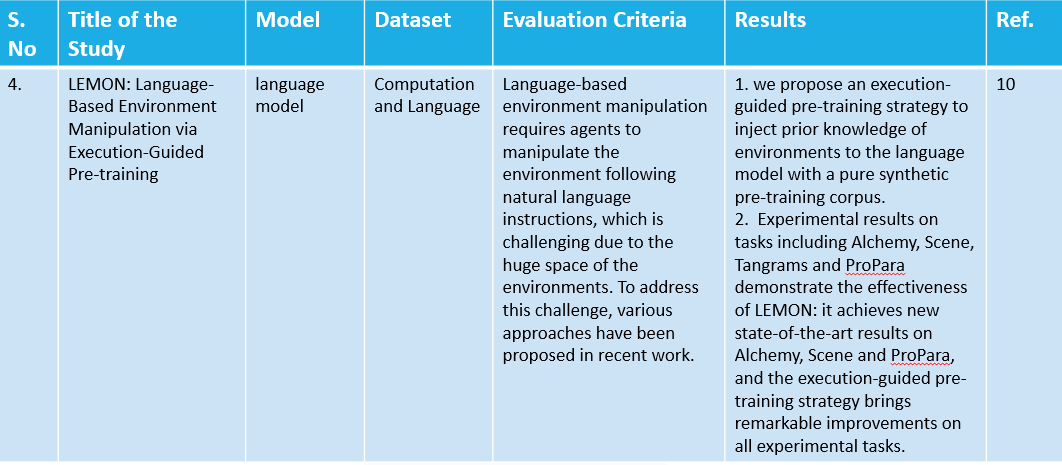
****

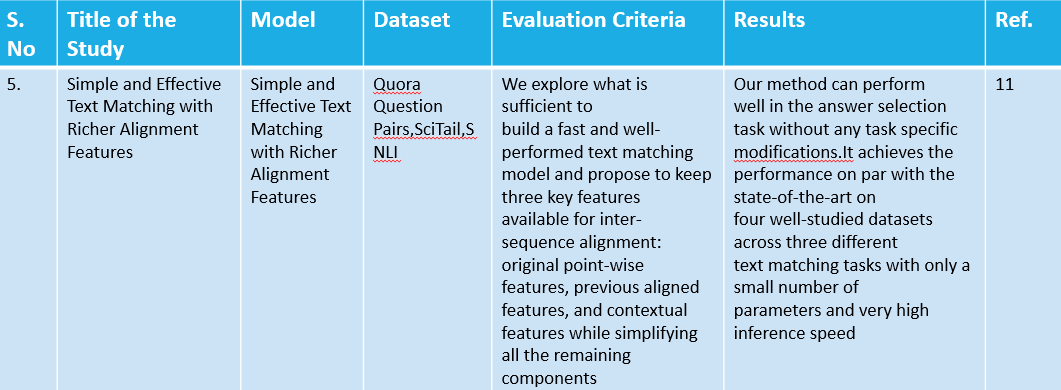
**Technique:**

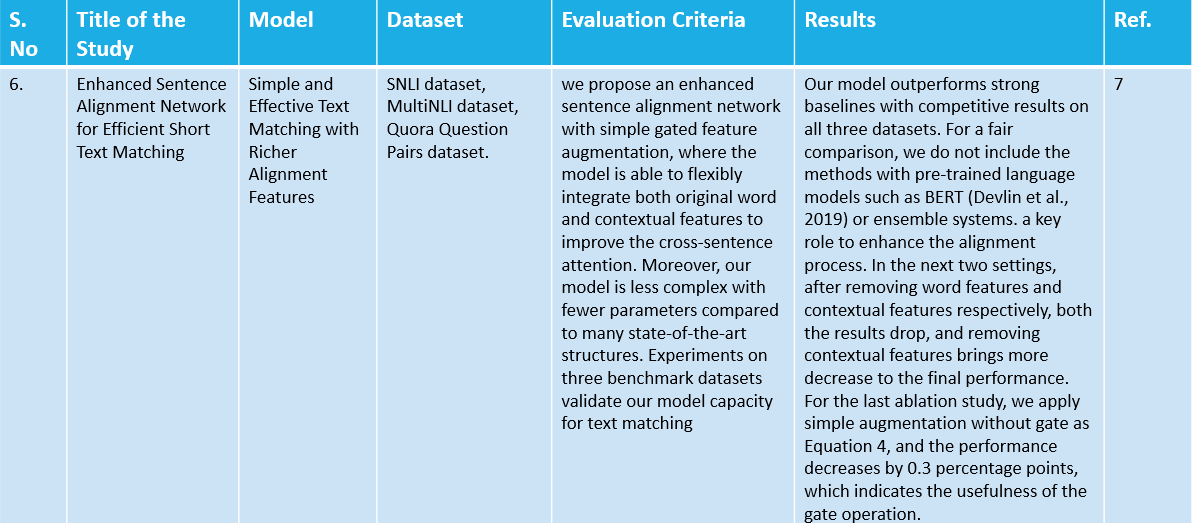
****

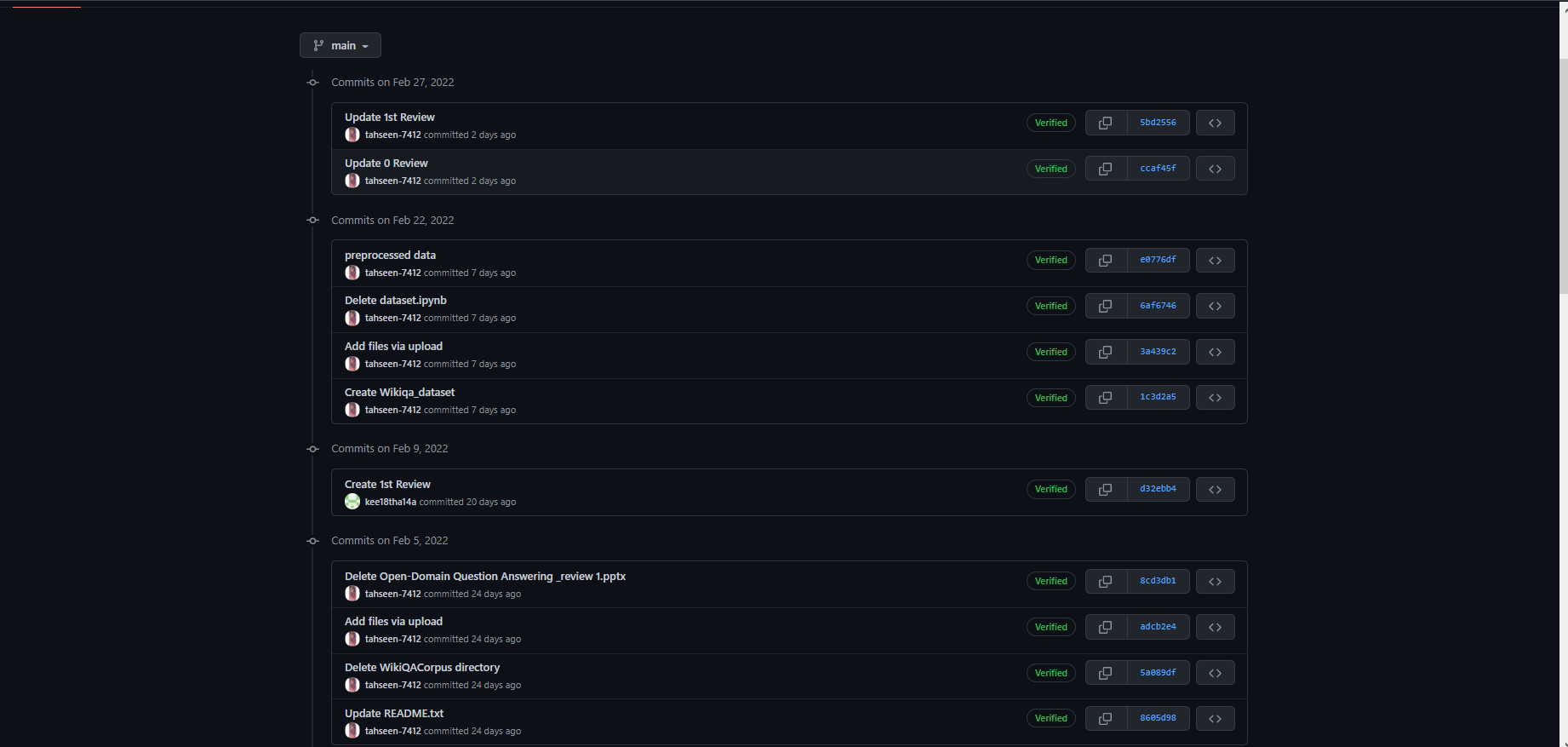
****

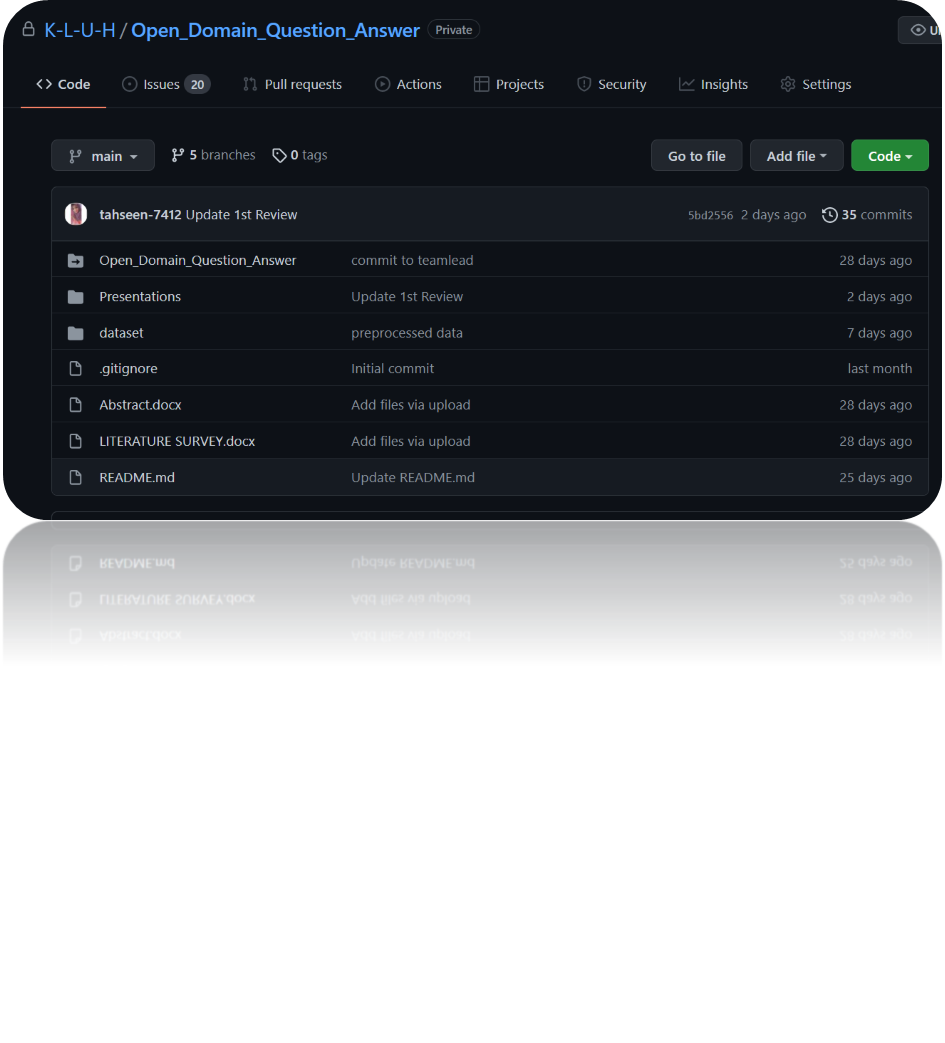
****

****

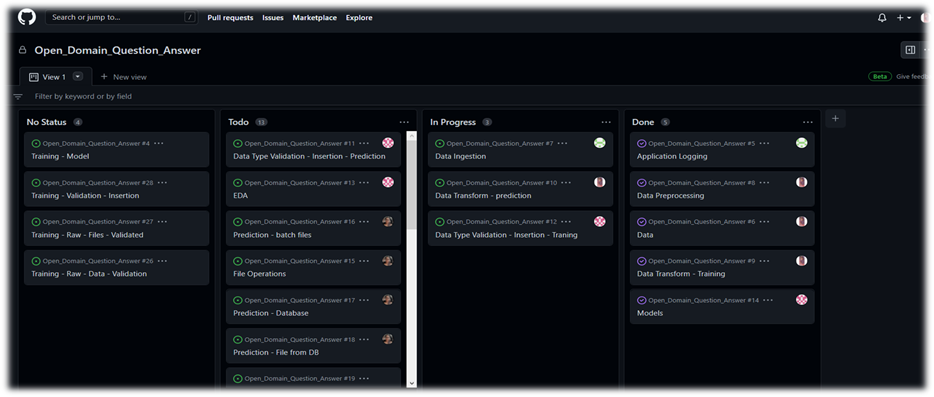
****

**Git commits (Interval 11 commits after 1st review)**

****

****

**Work In Progress**

****

**Pre-processing code and techniques**

step1:

!pip install datasets

step2:

from datasets import load\_dataset

squad = load\_dataset('squad\_v2',split = 'train')

Squad

step3:

print(squad[0]['context'][:60])

print(squad[0]['answers']['answer\_start'])

step4:

squad[0]['answers']['answer\_start'][0] + len(squad[0]['answers']['text'][0])

step5:

squad[0]['context'][269:286]

step6:

context = squad[0]['context']

gold\_text = squad[0]['answers']['text'][0]

start\_idx = squad[0]['answers']['answer\_start'][0]

end\_idx = start\_idx +len(gold\_text)

print(gold\_text)

print(start\_idx,end\_idx)

print(context[start\_idx:end\_idx])

step7:

squad[2075]

step8:

def get\_char\_positions(sample):

if len(sample['answers']['text']) == 0:

text = ''

answer\_start = 0

answer\_end = 0

else:

text = sample['answers']['text'][0]

answer\_start = sample['answers']['answer\_start'][0]

answer\_end = answer\_start + len(text)

return {

'text': text,

'answer\_start': answer\_start,

'answer\_end': answer\_end

}

squad = squad.map(lambda x: {

'answers': get\_char\_positions(x)

})

step9:

squad[0]

step10:

!pip install transformers

step11:

from transformers import BertTokenizerFast

tokenizer = BertTokenizerFast.from\_pretrained('bert-base-uncased')

step12:

squad = squad.map(lambda x: tokenizer(

x['question'], x['context'], max\_length = 384,

padding = 'max\_length', truncation = True,

return\_offsets\_mapping = True

))

step13:

squad[0]

step14:

tokenizer.decode(squad[0]['input\_ids'])

step15:

squad[0]['token\_type\_ids’]

step16:

question\_len = 0

for x in squad[0]['token\_type\_ids']:

if x != 1:

question\_len += 1

else:

break

context\_len = sum(squad[0]['token\_type\_ids'])

question\_len, context\_len

step17:

tokenizer.decode(squad[0]['input\_ids'][:question\_len])

step18:

tokenizer.decode(squad[0]['input\_ids'][question\_len:context\_len+question\_len])

step19:

context\_mappings = squad[0]['offset\_mapping'][question\_len:][:context\_len-1]

char\_start = squad[0]['answers']['answer\_start']

char\_end = squad[0]['answers']['answer\_end']

for i, mapping in enumerate(context\_mappings):

if char\_start >= mapping[0] and char\_start <= mapping [0]:

token\_start = question\_len + i

if char\_end >= mapping[0] and char\_end <= mapping[1]:

token\_end = question\_len + i + 1

break

if i == len(context\_mappings) - 1:

token\_start, token\_end = 0, 0

break

squad[0]['answers']['text']

step20:

squad

step21:

squad = squad.remove\_columns(['id','title','context','question','answers','offset\_mapping'])

squad

step22:

!pip install transformers

step23:

from transformers import BertForQuestionAnswering

model = BertForQuestionAnswering.from\_pretrained('bert-base-uncased')

step24:

from transformers import TrainingArguments

batch\_size = 24

epochs = 3

args = TrainingArguments(

'bert-base-uncased-squad2',

learning\_rate=2e-5,

per\_device\_eval\_batch\_size=batch\_size,

num\_train\_epochs=epochs,

weight\_decay=0.1,

warmup\_steps=int(len(squad) \* epochs \* 0.1)

)

from transformers import default\_data\_collator

data\_collator = default\_data\_collator

import torch

print(torch.cuda.is\_available)

from transformers import Trainer

import torch

device = 'cuda : 0' if torch.cuda.is\_available() else 'cpu'

trainer = Trainer(

model.to(device),

args,

train\_dataset = squad,

data\_collator = data\_collator,

tokenizer = tokenizer

)

trainer.train()

**Alpha Testing**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="with=device-width, initial-scale=1.0">

<title>Open Domain Question And Answer</title>

<link rel ="stylesheet" href="style.css">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

</head>

<body>

<section class="header">

<nav>

<div class="nav-links" id="navLinks">

<i class="fa fa-times-circle" onclick="hideMenu()"></i>

<ul>

<li><a href="index.html">HOME</a></li>

<li><a href="about.html">ABOUT</a></li>

<li><a href="QA.html">Q/A</a></li>

<li><a href="contact.html">CONTACT</a></li>

</ul>

</div>

<i class="fa fa-bars" onclick="showMenu()"></i>

</nav>

<div class="text-box">

<p>Open Domain Question And Answer</p>

<a href="" class="hero-btn">Click here to ask Question</a>

</div>

</section>

<!---------- call to action-------->

<center>

<section class="cta">

<h1>Team Contact</h1>

<a href="" class="hero-btn">CONTACT US</a>

</section>

<!----- footer ---------->

<section class="footer">

<h4>About Us</h4>

<p>Students from koneru lakshmaiah education foundation hyderabad</p>

<div class="icons">

<i class="fa fa-facebook"></i>

<i class="fa fa-twitter"></i>

<i class="fa fa-instagram"></i>

<i class="fa fa-linkedin"></i>

<i class="fa fa-github"></i>

</section>

</center>

<!-----JavaScript for Toggle Menu----->

<script>

var navLinks = document.getElementById("navLinks");

function showMenu(){

navLinks.style.right = "0";

}

function hideMenu(){

navLinks.style.right = "-200px";

}</script>

</body>

</html>



**Division of work among the group members**

Preprocessing code and Techniques – Tahseen Begum

Alpha Testing – E. Pravallika and N. Sowgna

Documentation, PowerPoint, Flowchart– N. Sowgna, Keerthana

**Conclusion**

We have completed the front-end for this project.

Next we will work on the back-end.

The question answering system to produce relevant, correct, and complete answers to the point.

Hence many evaluation metrics were developed to measure such ambiguous terminologies.

**References**

* Liu, Qian, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jian-guang Lou. "TAPEX: Table pre-training via learning a neural SQL executor." arXiv preprint arXiv:2107.07653 (2021).
* Hu, Zhe, Zuohui Fu, Cheng Peng, and Weiwei Wang. "Enhanced Sentence Alignment Network for Efficient Short Text Matching." In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pp. 34-40. 2020.
* Soldaini, Luca, and Alessandro Moschitti. "The cascade transformer: an application for efficient answer sentence selection." arXiv preprint arXiv:2005.02534 (2020).
* Shi, Qi, Qian Liu, Bei Chen, Yu Zhang, Ting Liu, and Jian-Guang Lou. "LEMON: Language-Based Environment Manipulation via Execution-Guided Pre-training." arXiv preprint arXiv:2201.08081 (2022).
* Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. Transactions of the Association for Computational Linguistics, 6:241–252.
* Guo, Zechen, Fucheng Wan, and Ning Ma. "Research and Implementation of Open Domain Question Answering System Based on DuReader Dataset and BIDAF Model." In Journal of Physics: Conference Series, vol. 1769, no. 1, p. 012033. IOP Publishing, 2021.
* Levy, Sharon, Kevin Mo, Wenhan Xiong, and William Yang Wang. "Open-Domain Question-Answering for COVID-19 and Other Emergent Domains." arXiv preprint arXiv:2110.06962 (2021).
* Zhan, Xunlin, Yinya Huang, Xiao Dong, Qingxing Cao, and Xiaodan Liang. "PathReasoner: Explainable reasoning paths for commonsense question answering." Knowledge-Based Systems 235 (2022): 107612.
* Lee, Chia-Hsuan, Shang-Ming Wang, Huan-Cheng Chang, and Hung-Yi Lee. "ODSQA: Open-domain spoken question answering dataset." In 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 949-956. IEEE, 2018.
* Yang, Yi, Wen-tau Yih, and Christopher Meek. "Wikiqa: A challenge dataset for open-domain question answering." In Proceedings of the 2015 conference on empirical methods in natural language processing, pp. 2013-2018. 2015.
* Herzig, Jonathan, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. "TaPas: Weakly supervised table parsing via pre-training." arXiv preprint arXiv:2004.02349 (2020).
* Moon, Sungrim, Huan He, Hongfang Liu, and Jungwei W. Fan. "RxWhyQA: a clinical question-answering dataset with the challenge of multi-answer questions." arXiv preprint arXiv:2201.02517 (2022).
* Antoniou, Christina, and Nick Bassiliades. "A survey on semantic question answering systems." The Knowledge Engineering Review 37 (2022).
* Hao, Tianyong, Xinxin Li, Yulan He, Fu Lee Wang, and Yingying Qu. "Recent progress in leveraging deep learning methods for question answering." Neural Computing and Applications (2022): 1-19.
* Yuan, Bowen, Bairu Chen, Zhiyi Tan, Xi Shao, and Bing-Kun Bao. "Unbiased feature enhancement framework for cross-modality person re-identification." Multimedia Systems (2022): 1-11.