Project 2

Question 1

Q1

For the dataset SQuAD2.0, it includes 3 impoortant parts:

1. Question: a string containing the question we will ask the model.
2. Context: a snippet of text that contains the answer to our question.
3. Answer: a shorter string that is an "excerpt" from the given context that provides the answer to our question.

step 1: We need to check if wether the answer should be extracted from *'answers'* or *'plausible_answers'*:
If the *'plausible_answers'* is in the *'qas'*: extract answer from *'plausible_answers'*, else extract answer from *'answers'*.

```
import json

def read_squad(path):
    # 打开JSON文件并加载字典
    with open(path, 'rb') as f:
        squad_dict = json.load(f)

    # 初始化上下文、问题和答案列表
    contexts = []
    questions = []
    answers = []
    # 在squad数据中迭代所有数据
    for group in squad_dict['data']:
        for passage in group['paragraphs']:
            context = passage['context']
            for qa in passage['qas']:
                question = qa['question']
                # 检查是否需要从'answers'或'plausible_answers'中提取
                if 'plausible_answers' in qa.keys():
                    access = 'plausible_answers'
                else:
                    access = 'answers'
                for answer in qa[access]:
                                    # 添加数据到列表
                                    contexts.append(context)
```

```
                                    questions.append(question)
                                    answers.append(answer)
    # 返回格式化的数据列表
    return contexts, questions, answers
```

Apply the function on the dataset:

```
train_contexts, train_questions, train_answers = read_squad('train-v2.0.json')
val_contexts, val_questions, val_answers = read_squad('dev-v2.0.json')
```

step 2: Randomly split the train dataset into train_set and val_set:

```
import random
sample_indexes = random.sample(range(len(train_contexts)), 5000)
val_contexts, val_questions, val_answers = \
     [train_contexts[i] for i in sample_indexes],[train_questions[i] for i in
sample_indexes],[train_answers[i] for i in sample_indexes]
```

step 3: The answer is contained in *"text"*, and the start of the answer in context is provided in *"answer_start"*. We need to train the model to find the beginning and end of an answer in context so we also need to add an *"answer_end"* value.

```
def add_end_idx(answers, contexts):
    # 循环每个answer-context对
    for answer, context in zip(answers, contexts):
        # gold_text指的是我们期望在上下文中找到的答案
        gold_text = answer['text']
        # 我们已经知道了起始索引
        start_idx = answer['answer_start']
        # #理想情况下，这将是结束索引...
        end_idx = start_idx + len(gold_text)

        # 然而，有时squad的答案会被一两个字符遗漏
        if context[start_idx:end_idx] == gold_text:
            # 如果答案不是off:
            answer['answer_end'] = end_idx
        else:
            # 这意味着答案相差1-2个标识
            for n in [1, 2]:
                if context[start_idx-n:end_idx-n] == gold_text:
                    answer['answer_start'] = start_idx - n
                    answer['answer_end'] = end_idx - n

add_end_idx(train_answers, train_contexts)
add_end_idx(val_answers, val_contexts)
```

step 3: Converts the string to an token, and then converts the answer start and answer end indexes from the character position to the token position.

- Initialize the tokenizer:

```
from transformers import DistilBertTokenizerFast

tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
# tokenize
train_encodings = tokenizer(train_contexts, train_questions, truncation=True,
padding=True)
val_encodings = tokenizer(val_contexts, val_questions, truncation=True,
padding=True)
```

- Converts the answer start and answer end indexes from the character position to the token position:

```
def add_token_positions(encodings, answers):
  # initialize lists to contain the token indices of answer start/end
  start_positions = []
  end_positions = []
  for i in range(len(answers)):
      # append start/end token position using char_to_token method
      start_positions.append(encodings.char_to_token(i, answers[i]
['answer_start']))
      end_positions.append(encodings.char_to_token(i, answers[i]['answer_end']))

      # if start position is None, the answer passage has been truncated
      if start_positions[-1] is None:
          start_positions[-1] = tokenizer.model_max_length
      # end position cannot be found, char_to_token found space, so shift one
token forward
      go_back = 1
      while end_positions[-1] is None:
          end_positions[-1] = encodings.char_to_token(i, answers[i]
['answer_end']-go_back)
          go_back +=1
  # update our encodings object with the new token-based start/end positions
  encodings.update({'start_positions': start_positions, 'end_positions':
end_positions})

# apply function to our data
add_token_positions(train_encodings, train_answers)
add_token_positions(val_encodings, val_answers)
```

step 4: Now we have the data ready and everything we need and we just need to convert it to the correct format for training with PyTorch.

- Construct a dataset object

```
import torch

class SquadDataset(torch.utils.data.Dataset):
    def __init__(self, encodings):
```

```
            self.encodings = encodings

    def __getitem__(self, idx):
            return {key: torch.tensor(val[idx]) for key, val in
self.encodings.items()}

    def __len__(self):
            return len(self.encodings.input_ids)

train_dataset = SquadDataset(train_encodings)
val_dataset = SquadDataset(val_encodings)
```

Q2

step 1:** Import and download DistilBertForQuestionAnswering from transformers.

```
from transformers import DistilBertForQuestionAnswering

model = DistilBertForQuestionAnswering.from_pretrained("distilbert-base-uncased")
```

step 2: Set up the PyTorch environment and initialize the DataLoader.

```
from torch.utils.data import DataLoader
from transformers import AdamW
from tqdm import tqdm

# 设置GPU / CPU
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
# 将模型移到被检测设备
model.to(device)
# 激活模型的训练模式
model.train()
# 初始化AdamW优化器的权重衰减
optim = AdamW(model.parameters(), lr=5e-5)

# 初始化训练数据的数据加载器
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
```

step 3: Load data and start fine-tuning the model.

```
for epoch in range(3):
    # 设置模型为训练模式
    model.train()
    # 设置循环(我们对进度条使用tqdm)
    loop = tqdm(train_loader, leave=True)
    for batch in loop:
        # 初始化计算的梯度(从上一步)
        optim.zero_grad()
        # 提取训练所需的所有张量批次
        input_ids = batch['input_ids'].to(device)
```

```
        attention_mask = batch['attention_mask'].to(device)
        start_positions = batch['start_positions'].to(device)
        end_positions = batch['end_positions'].to(device)
        # 训练模型，返回输出
        outputs = model(input_ids, attention_mask=attention_mask,
                        start_positions=start_positions,
                        end_positions=end_positions)
        # 提取损失
        loss = outputs[0]
        # 计算每个需要更新的参数的损失
        loss.backward()
        # 更新参数
        optim.step()
        # 在进度条上打印相关信息
        loop.set_description(f'Epoch {epoch}')
        loop.set_postfix(loss=loss.item())
```

```
/root/miniconda3/envs/sp/lib/python3.10/site-packages/transformers/optimization.py:391: FutureWarning: This implementation of AdamW is depre
cated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True
` to disable this warning
  warnings.warn(
Epoch 0:  39%|████      | 3201/8145 [1:27:26<2:27:00,  1.78s/it, loss=0.919]
```

step 4: Save the model and tokenizer.

```
model_path = 'models/distilbert-custom'
model.save_pretrained(model_path)
tokenizer.save_pretrained(model_path)
```

The next time load the model, you can use the saved model directly.

```
model = AutoModel.from_pretrained(model_path)
tokenizer = AutoTokenizer.from_pretrained(model_path)
```

step 6: Compute accuracy:

```
# switch model out of training mode
model.eval()

#val_sampler = SequentialSampler(val_dataset)
val_loader = DataLoader(val_dataset, batch_size=16)

acc = []

# initialize loop for progress bar
```

```
loop = tqdm(val_loader)
# loop through batches
for batch in loop:
    # we don't need to calculate gradients as we're not training
    with torch.no_grad():
        # pull batched items from loader
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        start_true = batch['start_positions'].to(device)
        end_true = batch['end_positions'].to(device)
        # make predictions
        outputs = model(input_ids, attention_mask=attention_mask)
        # pull preds out
        start_pred = torch.argmax(outputs['start_logits'], dim=1)
        end_pred = torch.argmax(outputs['end_logits'], dim=1)
        # calculate accuracy for both and append to accuracy list
        acc.append(((start_pred == start_true).sum()/len(start_pred)).item())
        acc.append(((end_pred == end_true).sum()/len(end_pred)).item())
# calculate average accuracy in total
acc = sum(acc)/len(acc)
```

step 5: bash script find the best parameter.

- python code

```
# get the hyperparameter from bash script
epoches = int(sys.argv[1])
batch_size = int(sys.argv[2])

#...

# return accuracy back to bash script
sys.exit(acc)
```

- bashcript

```
echo "Start Training"

epoch_lo=1
epoch_up=5
batch_lo=10
batch_up=20
results=()
max_value=-1
best_epoch=1
best_batch=10

for ((param1=$epoch_lo; param1<=$epoch_up; param1++)); do
    for ((param2=$batch_lo; param2<=$batch_up; param2++)); do

        # 运行 Python 模型并传递参数
```

```
        #echo "$param1"
        acc=$(python Q12.py "$param1" "$param2")
        results+=("$acc")
        if (( acc > max_value )); then
            max_value=$acc
            best_epoch="$param1"
            best_batch="$param2"
        fi

    done
done

echo "best acc: $max_value, corresponding epoch: $best_epoch, corresponding batch:
$best_batch"
```

Q3

1. start kafka as before, and create "producer.py" to input our data into stream.(the data is transformed from Q1 and saved as .csv, in this section, I directly input processed csv file)

```
producer = KafkaProducer(bootstrap_servers='localhost:9092')
def read_data_and_send_messages():
    with open(r'/data/jupyter-data/project2/val.csv', 'r') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            message = {
                "question": (row["question"]),
                "context": (row["context"]),
                "answer": (row["answer"])}
            #this is all we need to fit a model
            producer.send('q3', json.dumps(message).encode('utf-8'))
            time.sleep(0.5)
```

2. Download the model as spark-nlp/HuggingFace in Spark NLP - DistilBertForQuestionAnswering.ipynb at master · JohnSnowLabs/spark-nlp (github.com). And deploy it as nlp-spark.

and load the model we just download :

```
MODEL_NAME = 'distilbert-base-cased-distilled-squad'

document_assembler = MultiDocumentAssembler() \
    .setInputCols(["question", "context", "true_ans"]) \
    .setOutputCols(["document_question", "document_context", "document_ans"])
```

```
spanClassifier_loaded =
DistilBertForQuestionAnswering.load("./{}_spark_nlp".format(MODEL_NAME))\
  .setInputCols(["document_question",'document_context'])\
  .setOutputCol("answer")

pipeline = Pipeline().setStages([
    document_assembler,
    spanClassifier_loaded
])
```

3. get the stream data from kafka

```
# 定义输入流，schema包含context和question两列
input_schema = StructType([
    StructField("context", StringType()),
    StructField("question", StringType()),
     StructField("answer", StringType())])
stream = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "q3") \
    .option("startingOffsets", "latest") \
    .load() \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json("value", input_schema).alias("input")) \
    .select("input.*")
```

4. deploy the model to the stream data, use dataframe to fit the model and get the result.

```
result = pipeline.fit(df1).transform(df1)
df2 = result.select("question", "context", "true_ans", "answer.result")
df2.show(30)
```

question	context	true_ans	result
In what country i...	The Normans (Norm...	France	[France]
When were the Nor...	The Normans (Norm...	10th and 11th cen...	[10th and 11th ce...
From which countr...	The Normans (Norm...	Denmark, Iceland ...	[Denmark , Icelan...
Who was the Norse...	The Normans (Norm...	Rollo	[Rollo]
What century did ...	The Normans (Norm...	10th century	[10th]
Who was the duke ...	The Norman dynast...	William the Conqu...	[William the Conq...
Who ruled the duc...	The Norman dynast...	Richard I	[Richard I]
What religion wer...	The Norman dynast...	Catholic	[Catholic]
What is the origi...	The English name ...	Viking	[Norseman , Viking]
When was the Lati...	The English name ...	9th century	[9th century]
When was the Duch...	In the course of ...	911	[911]
Who did Rollo sig...	In the course of ...	King Charles III	[King Charles III...
What river origin...	In the course of ...	Seine	[Seine]
Who upon arriving...	Before Rollo's ar...	Rollo	[Viking]
What was the Norm...	The descendants o...	Catholicism	[Catholicism]
What part of Fran...	The descendants o...	north	[north]
What was one of t...	The Normans there...	fighting horsemen	[fighting horsemen]
Who was the Norma...	Soon after the No...	Seljuk Turks	[Pechenegs , the ...
When did Herve se...	One of the first ...	1050s	[1050s]
When did Robert C...	One of the first ...	1060s	[1060s]
Who ruined Rousse...	One of the first ...	Alexius Komnenos	[Alexius Komnenos]
What was the name...	Some Normans join...	Afranji	[Afranji]
Who was the leade...	Some Normans join...	Oursel	[Oursel]
Who did the Norma...	Some Normans join...	Turkish forces	[Armenian state]
What were the ori...	Several families ...	Norman mercenary	[The Raoulii were...
What was the name...	Robert Guiscard, ...	Robert Guiscard	[Robert Guiscard]

5. **I also tried some other method**. Directly deploy use the model from "transformers"(we can get scores from this model)

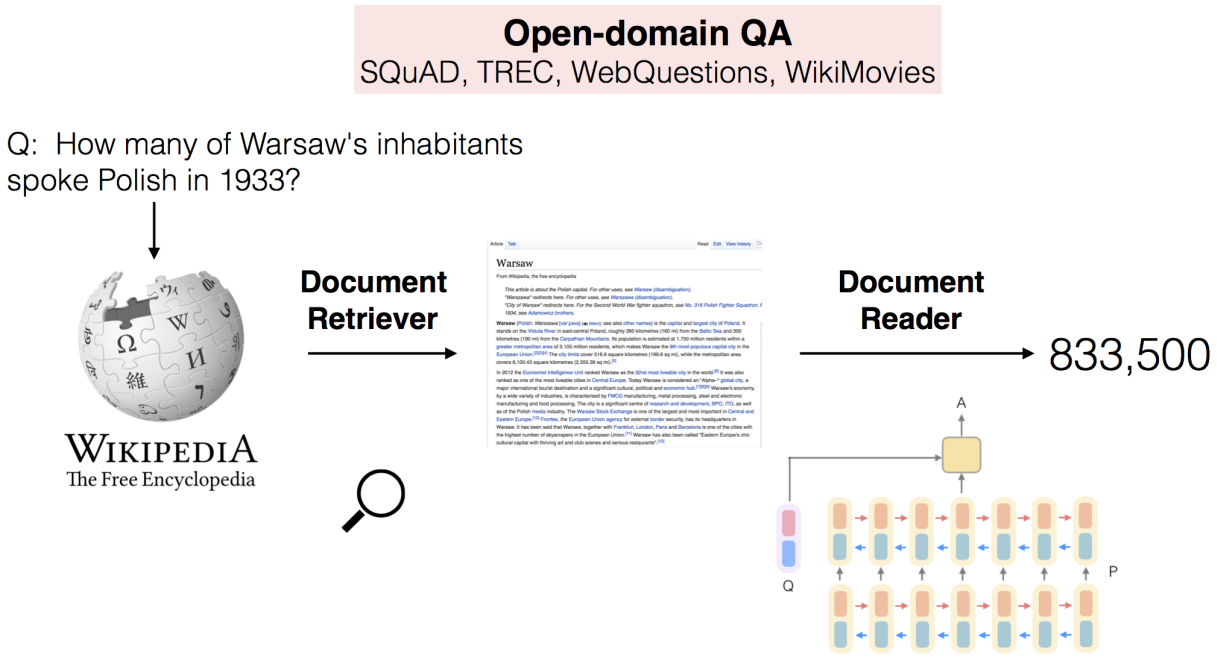
```
from transformers import pipeline
question_answerer = pipeline("question-answering", model='distilbert-base-cased-distilled-squad')
```

question	answer	model_answer	score
What happened to ...	steep and steady ...	A steep and stead...	0.4914613366127014
What gained groun...	anti-democratic I...	different democra...	0.4869297444820404
What gained groun...	anti-democratic I...	different democra...	0.4869297444820404
Who was the ideol...	Ali Shariati	Ali Shariati	0.9576330184936523
Mohammad Iqbal wa...	ideological	ideological	0.9344263672828674
Where does Khomei...	somewhere between	somewhere between	0.41264471411705017
Where does Khomei...	between	somewhere between	0.41264471411705017
Who was it essent...	the Prophet Mohammad	Prophet Mohammad	0.43677428364753723
Who was it essent...	Prophet Mohammad ...	Prophet Mohammad	0.43677428364753723
Who was it essent...	Prophet Mohammad	Prophet Mohammad	0.43677428364753723
What long term ag...	conspiracy	conspiracy agains...	0.45556145906448364
What long term ag...	Westernizing Muslims	conspiracy agains...	0.45556145906448364

Q4

1. Introduction

DrQA is a system for reading comprehension applied to open-domain question answering. In particular, DrQA is targeted at the task of "machine reading at scale" (MRS). In this setting, we are searching for an answer to a question in a potentially very large corpus of unstructured documents (that may not be redundant). Thus the system has to combine the challenges of document retrieval (finding the relevant documents) with that of machine comprehension of text (identifying the answers from those documents).



DrQA is mainly composed of Document Retriever and Document Reader. Document Retriever Retrieves related articles in Wikipedia based on the question and Document Reader finds the corresponding paragraph in the passage and extracts the answer.

2. Document Retriever

- **Document representation**
  Parse the document into the word bag of unigrams/bigrams, and then use hash feature to convert the word bag of length of 2^{24}, and then convert it into the TF-IDF vector. Splicing the vectors of all the documents together will turn it into a matrix. Each element in the matrix represents the frequency of word i in document j. This is a very sparse matrix, so the Sparse matrix (sp, sparse) module of scipy was used to save the data.
- **Question representation**
  Similar to the document representation, put the question text, convert the question text to TF-IDF vector.
- **Question retrieval**
  Calculate the similarity score of the question and each document and returne the most similar k documents.

3. Document Reader

- **Paragraph encoding**
  Each word in the paragraph is represented by its feature vector, which consists of the following four parts:

1. **Word embeddings**
   Using the pre-trained 300-dimensional Glove word vector, most of it remained unchanged, fine-tuning only 1000 words with the highest frequency in the question text. Because some keywords, such as what, how, which, etc., may be important for QA systems.
   $f_{emb}\left(p_i\right)=E\left(p_i\right)$
2. **Exact match**
   Use three simple binary features, indicating whether p_i can be exactly matched to one question word in q, either in its original, lowercase or lemma form.
   $f_{\text{exact-match}}\ (pi)=I\left(pi \in q\right)$
3. **Token features**
   Add a few manual features which reflect some properties of token pi in its context, which include its part-of-speech (POS) and named entity recognition (NER) tags and its (normalized) term frequency (TF).

$$f_{\text{token}}\ \left(p_i\right)=\left(\text{POS}\left(p_i\right), \text{NER}\left(p_i\right), TF\left(p_i\right)\right)$$

4. **Aligned question embedding**
   Use the attention mechanism to reflect the relevance of a word in a paragraph to each word in the question.

$$a_{i,j}=\frac{\exp\left(\alpha\left(\mathbf{E}\left(p_i\right)\right) \cdot \alpha\left(\mathbf{E}\left(q_j\right)\right)\right)}{\sum_{j'} \exp\left(\alpha\left(\mathbf{E}\left(p_i\right)\right) \cdot \alpha\left(\mathbf{E}\left(q_{j'}\right)\right)\right)}$$

   $\alpha\left(.\right)$ is a single dense layer with ReLU nonlinearity. Compared to the exact match features, these features add soft alignments between similar but non-identical words.

   Represent all tokens pi in a paragraph p as a sequence of feature vectors $\tilde{\mathbf{p}}_i \in \mathbb{R}^d$ and pass them as the input to a recurrent neural network and thus obtain:

$$\left\{p_1, \ldots, p_m\right\}=\text{RNN}\left(\left\{\tilde{p}_1, \ldots, \tilde{p}_m\right\}\right)$$

   The state vector for each LSTM cell in each hidden layer is taken as output.

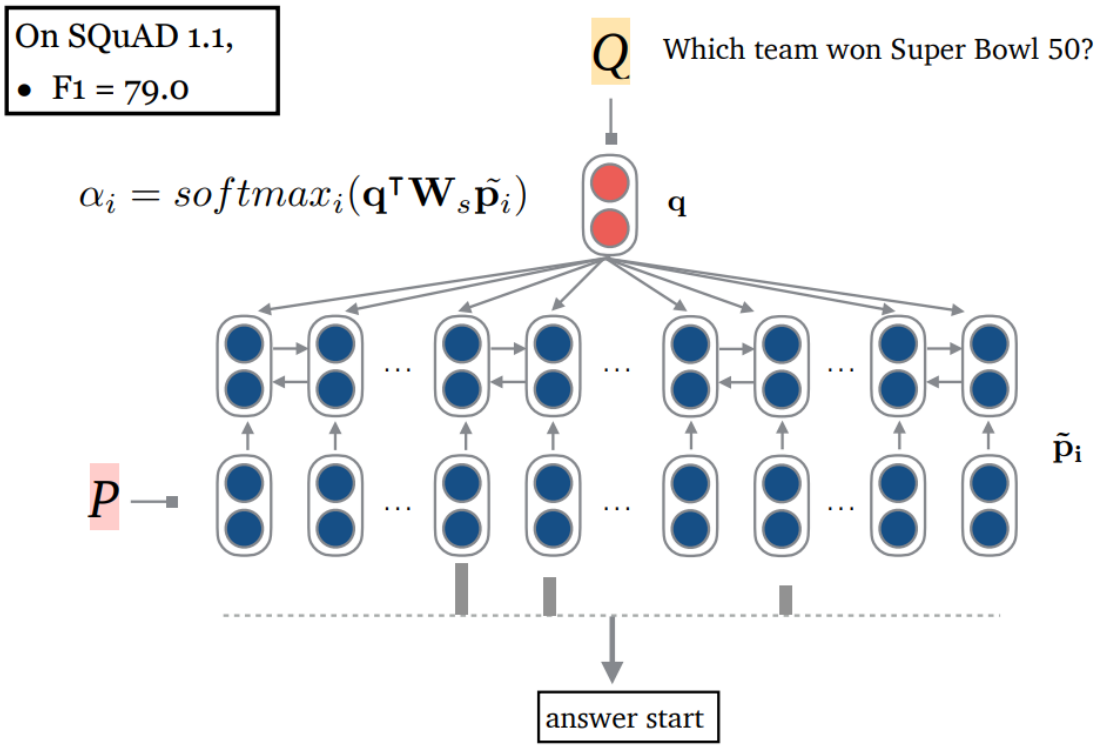
- **Question encoding**
  Use another LSTM to encode the embedding of every word and it could get $\left\{q_1, \ldots, q_l\right\}$. Weighting and summing the vectors of these hidden layer outputs yields the expression vector $\mathbf{q}=\sum_j b_j \mathbf{q}_j$ of a problem, where b_j encodes the importance of each question word and $\mathbf{w}$ is a weight vector to learn.

$$b_j=\frac{\exp\left(\mathbf{w} \cdot \mathbf{q}_j\right)}{\sum_{j'} \exp\left(\mathbf{w} \cdot \mathbf{q}_{j'}\right)}$$

- **Prediction**
  Use paragraph vectors $\{q_1, \ldots, q_l\}$ and sum problem vector q as input, for each position i of the paragraph, predict the probability that it will be the starting and ending position of the answer:

$$P_{\text{start}}(i) \propto \exp(p_i W_s q)$$
$$P_{\text{end}}(i) \propto \exp(p_i W_e q)$$

```
Use $softmax\left(PWQ\right)$ as the output $P_{start}$ and $P_{end}$.
```



Question 2

DeepMind

DeepMind is a UK-based artificial intelligence company co-founded by AI programmer and neuroscientist Demis Hassabis and others in 2010. It was originally named DeepMind Technologies Limited and was acquired by Google in 2014. The company combines the most advanced technologies in machine learning and systems neuroscience to build powerful general-purpose learning algorithms that will initially be used in business areas such as simulation, e-commerce, game development, and eventually healthcare.

Achievement

1. **AlphaGo**: In 2016, DeepMind's AI program AlphaGo defeated the world champion of the ancient Chinese board game Go. This was the first time that an AI program had beaten a human professional player at Go, which was considered a significant milestone in the development of AI.
2. **AlphaZero**: In 2017, DeepMind developed AlphaZero, an AI program that learned to play Go, chess, and shogi (a Japanese board game) at a superhuman level using a single algorithm and without any prior knowledge of the games. AlphaZero's ability to master complex games using a general learning algorithm was seen as a major breakthrough in AI research.
3. **Protein folding**: In 2020, DeepMind's AI system AlphaFold made a significant breakthrough in the field of protein folding prediction. AlphaFold was able to predict the structure of a protein with high accuracy, which is a critical step in understanding how proteins work and developing new drugs.
4. **MuZero**: In 2020, DeepMind developed MuZero, an AI program that can learn to play board games, video games, and other tasks without any prior knowledge of the rules. MuZero's ability to learn through trial and error was seen as a major step forward in the development of more general AI systems.

   .....

Based on above achievement, we can see why DeepMind is widely regarded as one of the most innovative and influential companies in the AI industry. The company has been at the forefront of several breakthroughs in AI research and has produced some of the most advanced AI systems in the world.

Strengths

1. Strong team of researchers and engineers

According to Google Scholar, as of May 2023, DeepMind researchers have published over 1000 research papers, including many in top-tier AI conferences such as NeurIPS, ICML, and ICLR. Publications (deepmind.com)

Another indicator of DeepMind's research strength is the number of citations its papers have received. According to the same source, DeepMind papers have been cited over 64,000 times, with an average citation per paper of over 60.

2. Partnerships with leading companies

For example, in healthcare, DeepMind has partnered with the UK's National Health Service (NHS) to develop AI systems for diagnosing and treating diseases. And other companies like Google, NHS, Unilever, NASA, the Montreal Institute for Neurocomputing, and so on. In finance, DeepMind has partnered with several major banks to develop AI systems for fraud detection and risk management. And in energy, DeepMind has partnered with several energy companies to develop AI systems for optimizing energy usage and reducing carbon emissions.

3. Values

"Our values are the driving force behind everything we do and our success depends on staying true to them." This is a quote on the DeepMind homepage. The top 5 most important values are:

1. Mission Driven: the team has the same drive which is to solving intelligence, to advance science and benefit humanity.
2. Pioneering: they are focusing on works which are groundbreaking and inspiring, and their goals are incredibly ambitious.
3. Responsible: they have the responsibility to bring this powerful technology into the world, and ethical considerations have to be at the forefront of all of their work.
4. Collaborative: the diversity of experience, knowledge, background and perspectives that the team have are flexible in the way they behave and work as a team.
5. Kind: The team are loyal and protective, seeing the very best in each other and always ready to give the benefit of the doubt.

I believe it is the corporate culture that has made this team so powerful and efficient, which leads them to make breakthroughs in AI.

Weakness

As a private company, DeepMind does not publicly disclose any weaknesses or shortcomings it may face. However, there are some potential challenges that could be viewed as weaknesses for the company:

1. **High costs**: DeepMind invests heavily in research and development, which can be expensive. As a result, the company may face financial challenges as it seeks to develop and scale new AI technologies.
2. **Privacy concerns**: DeepMind has faced criticism and legal challenges over its handling of patient data in its healthcare partnerships with the UK's National Health Service. As the company expands its work in healthcare and other industries, it may face additional scrutiny over its data practices.
3. **Ethical concerns**: As an AI company, DeepMind faces questions about the potential impact of its technology on society and the broader ethical implications of its work. The company has established an ethics unit to address these concerns, but it remains to be seen how successful it will be in navigating these complex issues.

(But I think these are also the problems for all AI companies, so I don't think they will put DeepMind into disadvantages)

Opportunities

1. **Advancements in computing technology**: DeepMind's work relies heavily on advances in computing technology, particularly in areas like machine learning and neural networks. The rapid development of these technologies over the past decade has created new opportunities for DeepMind to develop more powerful AI systems.
2. **Increased availability of data**: The rise of big data and the growth of the internet have led to a massive increase in the amount of data available for analysis. This has created new opportunities for DeepMind to develop AI systems that can learn from large datasets and make more accurate predictions and decisions.
3. **Supportive regulatory environment**: In the UK, where DeepMind is based, the government has been supportive of AI research and development. This has created a favorable regulatory environment for the company, making it easier to conduct research and collaborate with other organizations.
4. **Growing demand for AI solutions**: Across industries, there is a growing demand for AI solutions that can help organizations automate tasks, make better predictions, and improve decision-making. This has created new opportunities for DeepMind to develop and deploy its AI systems in a range of settings.

Threats

1. **Increased competition**: The AI industry is highly competitive, with many large tech companies investing heavily in AI research and development. As more companies enter the market, DeepMind may face increased competition for talent and resources, as well as pressure to stay ahead of its competitors in terms of innovation and market share.
2. **Regulatory challenge**: The use of AI is subject to regulation in many industries, and regulatory bodies are still grappling with how to address the ethical and safety concerns posed by AI. As a result, DeepMind may face regulatory challenges that could limit its ability to operate in certain industries or geographies.
3. **Public scrutiny**: As an AI company, DeepMind faces public scrutiny and concerns about the potential impact of its technology on society. Any negative public perception of the company or its work could harm its reputation and limit its ability to collaborate with partners or secure funding.
4. **Cybersecurity threats**: The use of AI systems may make organizations more vulnerable to cybersecurity threats, including data breaches and cyber attacks. DeepMind's work with large datasets and sensitive information could make it a target for cybercriminals seeking to steal valuable data.

   Overall, these threats are not unique to DeepMind and are common among many companies operating in the AI industry. However, they represent potential challenges that the company may need to navigate in order to continue to thrive and grow.

Strength	Weakness	Opportunities	Threats
Strong team of researchers and engineers	High costs	Advancements in computing technology	Increased competition

Strength	Weakness	Opportunities	Threats
Partnerships with leading companies	Privacy concerns	Increased availability of data	Public scrutiny
Core Values	Ethical concerns	Supportive regulatory environment	Cybersecurity threats
		Growing demand for AI solutions	

Synthesis AI

Synthesis AI is a company that creates realistic digital humans for AR, VR and Metaverse applications. They are the first company to demonstrate text-to-3D digital human synthesis at such a high level of quality and detail. The technology allows users to input text descriptions of the desired digital human, such as age, gender, ethnicity, hairstyle and clothing, then generate a 3D model that matches the specifications. Synthesis AI also provides synthetic data for computer vision to enable more capable and ethical AI.

Achievements

Synthesis AI's products include **HumanAPI** which is an on-demand generation of diverse, photoreal, labeled facial and full-body data to support AI model development. Synthesis AI is also the technology leader for creating realistic digital humans. Its digital human datasets support the many different types of avatar characteristics, including skin tones, gestures, and facial expressions, needed for next-generation virtual innovation in the Metaverse.

In April, Synthesis AI announces that it has developed a Generative AI Avatar scheme "3D Generative AI" that enables the creation of realistic digital human beings through text generative prompts. The technology will be available to a select group of testers starting in the second quarter of this year. 3D Generative AI uses generative AI and visual effects pipelines to produce high-resolution, high-quality virtual digital humans that can be used for a variety of applications, including games, VR, film and simulation, the company said.

Strength

1. **User-friendly interface**: Synthesis AI has developed a user-friendly interface that allows users to easily create and customize AI models without needing extensive programming knowledge. This ease of use could make it appealing to a broader range of users and businesses.
2. **synthetic data**: AI-controlled synthesis has the following advantages: (1) safety; (2) high speed; (3) reproducibility; and (4) low cost of synthesizing compounds. The benefits of synthetic data include on-demand labeled images and videos, highly scalable data generation platform that delivers millions of perfectly labeled images. The use of synthetic data can help businesses and organizations to overcome some of the challenges associated with traditional data collection and management, including privacy concerns, cost, scalability, and quality control.
3. **AI-powered content optimization**: Synthesis AI's platform uses AI to optimize content for specific goals, such as increasing engagement or improving search engine rankings. This could make it appealing to businesses looking to improve their online presence and content marketing efforts.

Weakness

1. **New entrant**: Synthesis AI is a relatively new company and does not have the same brand recognition or market share as more established players in the AI space.
2. **Limited offerings**: Synthesis AI's platform is currently focused on humanAPI and content optimization, which could limit its appeal to businesses and organizations looking for AI solutions in other areas.

Opportunities

1. **Growing demand for synthetic data**: The demand can be attributed to several factors, including privacy concerns, limited data availability, data bias, cost and scalability, diversity, and the COVID-19 pandemic. These factors have led to an increasing interest in synthetic data across various industries. According to a report by MarketsandMarkets, the global synthetic data market is expected to grow at a CAGR of 39.9% from USD 178 million in 2020 to USD 959 million by 2025..
2. **Expansion into other areas of AI**: Synthesis AI could expand its offerings to include other areas of AI, such as computer vision or predictive analytics, in order to appeal to a broader range of businesses and organizations.

Threats

1. **Competition from established players**: Synthesis AI faces competition from more established players in the AI space, such as Google, Microsoft, and IBM.
2. **Privacy concerns**: As with any company working with data, Synthesis AI must be careful to protect user privacy and comply with data protection regulations.
3. **Rapidly changing technology**: The field of AI is constantly evolving, which means that Synthesis AI will need to keep up with new developments and technologies in order to remain competitive.

Source: Compiled by the AI language model, ChatGPT, based on its general knowledge of Synthesis AI and the AI industry as of its knowledge cutoff in September 2021.

Strength	Weakness	Opportunities	Threats
User-friendly interface	New entrant	Growing demand for synthetic data	Competition from established players
advantages of synthetic data	Limited offerings	Expansion into other areas of AI	Privacy concerns
AI-powered content optimization			Rapidly changing technology

What I learned from these two startup?

DeepMind's core value impressed me most sources: Careers (deepmind.com)

Are there any similar opportunities in China?

There are similar opportunities in China. China has a growing AI industry and many companies are investing heavily in AI research and development. Some Chinese companies are also working on developing advanced AI technologies similar to those developed by DeepMind.

For example, Alibaba's research arm, the Alibaba DAMO Academy, has been working on developing advanced AI technologies such as natural language processing and computer vision. Another Chinese company, Tencent, has also been investing in AI research and development, and has developed its own AI-powered medical imaging platform.

In addition, there are many startups in China that are focused on AI and data science, such as SenseTime, which develops facial recognition technology, and Megvii, which develops AI-powered visual recognition technology.

However, it is worth noting that the Chinese AI industry operates in a different regulatory environment than the West, and there have been concerns about the use of AI for surveillance and other potentially harmful applications. Therefore, while there are similar opportunities in China for AI development, there are also unique challenges and risks associated with operating in this market.

Some Drawbacks of ChatGPT/New Bing

I mainly use ChatGPT and New Bing in this report. Although they are really convenient, there are still a lot of shortcomings.

1. **Wrong information**: When I asked ChatGPT about the publication numbers of DeepMind, it quickly give me an answer. But I asked it again, then I found that the answers are different. So I open the official website of DeepMind and found that ChatGPT just made up an answer to satisfy my need.
2. **Wrong/No sources**: Every time I asked ChatGPT, I will require it to give the source of the information or links, but almost half of the time it just gave me a link that could not open at all! Compared to ChatGPT, New bing is more reliable,which can give the right sources. But sometimes it won't give the links or sources if it can't find any.
3. **Too general answer:** sometimes I want concrete and specific answer with examples to illustrate. However, most of the time, AI's answers are too general. For instance, I ask AI: "what's the drawbacks of DeepMind compared to other startup?" , then it gives the answer which is suitable for all AI startups, no matter how I asked.

Contribution

Yicheng Wu Question 1.1/ 1.2/ 1.4

Jiawei Xiong Question 1.3 / Question 2

Reference (webpage)

1. Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading Wikipedia to Answer Open-Domain Questions (arXiv:1704.00051). arXiv
2. huggingface/transformers: 🤗 Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX. (github.com)
3. The Stanford Question Answering Dataset (rajpurkar.github.io)
4. danqi/acl2020-openqa-tutorial: ACL2020 Tutorial: Open-Domain Question Answering (github.com)
5. Careers (deepmind.com)
6. Publications (deepmind.com)
7. Synthetic Data for Computer Vision - Synthesis AI