Reading Comprehension Backend - AI Core

Platforms and Tools

- **Flask** - python
- **MongoDB** - pymongo
- **RabbitMQ** - pika
- **AllenNLP** - rc
- **Swagger**
- **Docker**

Deployment

server = 192.168.55.215

+Access Root of project file: $ cd ~/RC

+Build backend-services (mongo, rabbitmq, data_controller, train_service, infer_service):

```
docker-compose build
docker-compose up -d
```

+ Build AI-service (train_worker, infer_worker) from: https://hub.docker.com/u/noahdrisort

```
docker pull noahdrisort/rc_worker

docker build --tag rc_worker:latest InferService/worker
```

```
docker run --detach --name rc_worker_infer_1 --gpus all \\
       -e SERVER_IP='192.168.55.215' -v ${PWD}/Resource:/app/Resource rc_worker:latest

docker run --detach --name rc_worker_train_1 --gpus all \\
       -e SERVER_IP='192.168.55.215' -v ${PWD}/Resource:/app/Resource rc_worker:latest
```

+Init sample database:

http://server:6006/datainit

+Wait 30s for AI-model and rabbitMQ init

+Start using API call

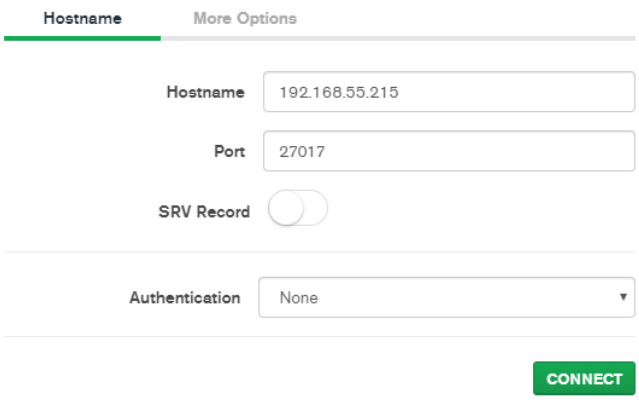
- For short build: (build image and run container)
  - $ chmod +x build.sh
  - $ ./build.sh
- For short delete: (stop container and remove image)
  - $ chmod +x delete.sh
  - $ ./delete.sh
- For short up: (start containers)
  - $ chmod +x up.sh
  - $ ./up.sh
- For short down: (stop containers)
  - $ chmod +x down.sh
  - $ ./down.sh

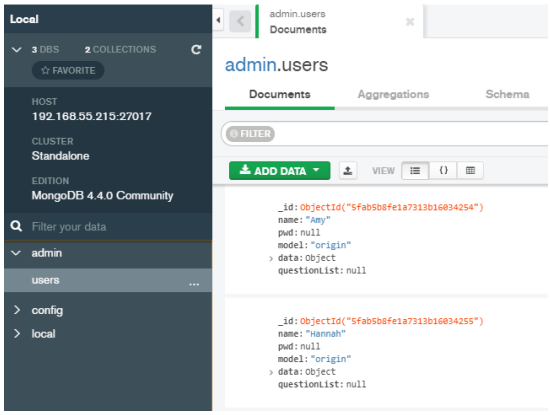
Management

On Mongo Compass

* View Mongo:

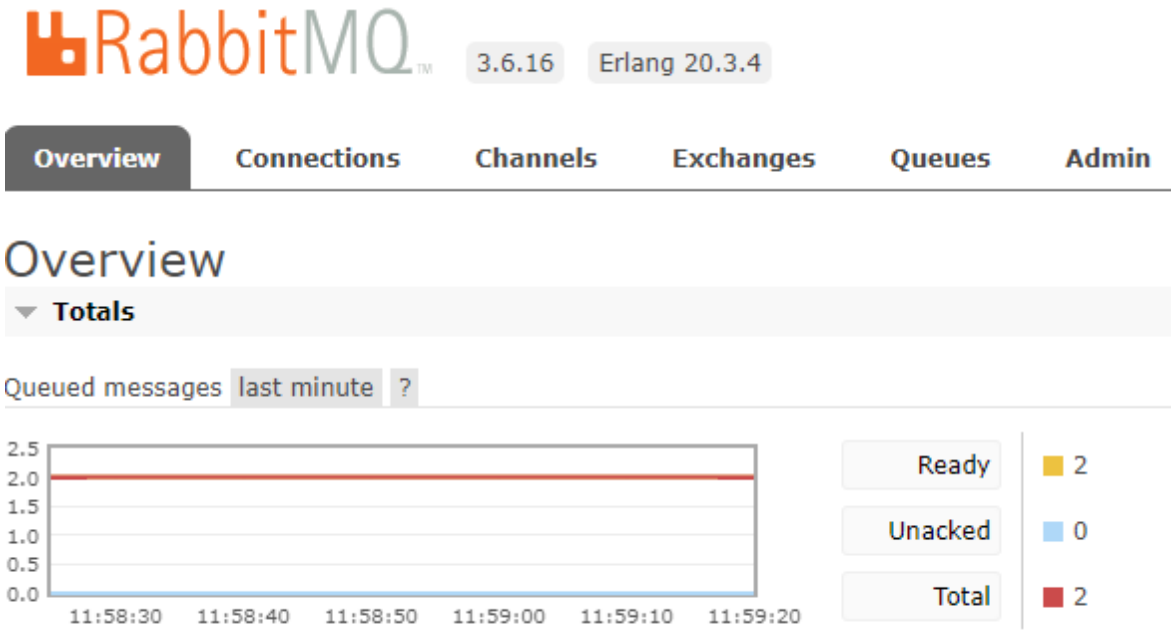
mongodb://server:27017/?readPreference=primary&appname=MongoDB%20Compass





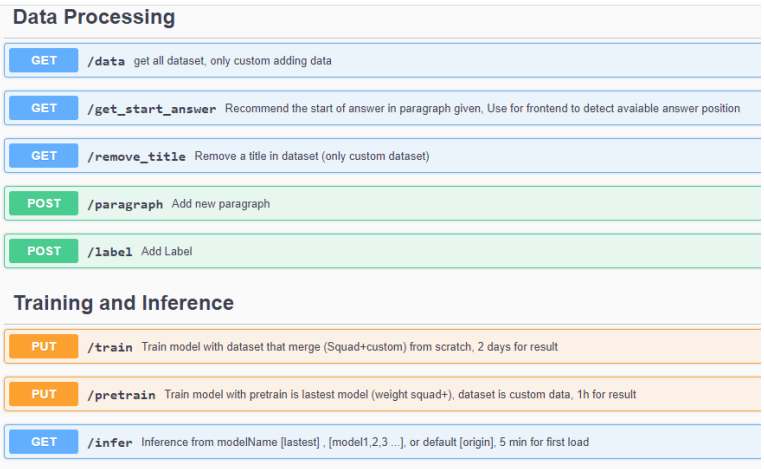
On Browser

* View Rabbit: http://server:15672



Overview			Messages			Message rates		
Name	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
infer_queue	D	idle	2	0	2	0.00/s	0.00/s	0.00/s
res_infer_queue		idle	0	0	0			
res_train_queue		idle	0	0	0	0.00/s	0.00/s	0.00/s
train_queue	D	idle	0	1	1	0.00/s	0.00/s	0.20/s

* View Swagger: http://server/swagger



On Server terminal

* View container logs:

$ docker logs -f {container_id} (get container id by: docker ps)

List of service:

- mongo: 27017
- rabbitmq: 5672
- rc_server_train: 5001
- rc_server_infer: 5002
- rc_controller: 6006
- rc_worker_train
- rc_worker_infer

* Access container:

$ docker exec -it {container_id} /bin/bash

Using API call

API should run on server with GPU

Data_service: server/6006

```
**Get full data**
path: /data
Description: get all dataset SQUAD+
methods=['GET']
Required param: None

**Recommend Start-answer from context and answer**
path: /get_start_answer
Description: Use for frontend to detect avaiable answer position
methods=['GET']
Required param: context , answer

**Remove a title from dataset**
path: /remove_title
Description: remove a title from dataset
methods=['GET']
```

```
Required param: title


**Add Paragraph**
path: /paragraph
Description: Add new paragraph of a title to current dataset, if title is
not exist, add new title
methods=['POST']
Body form: title , context

**Add Label**
path: /label
Description: Add label for question Q, paragraph P in title T
methods=['POST']
Body form: title , context , question , answer , start (start is position
of answer in context)
```

Train_service: server/5001

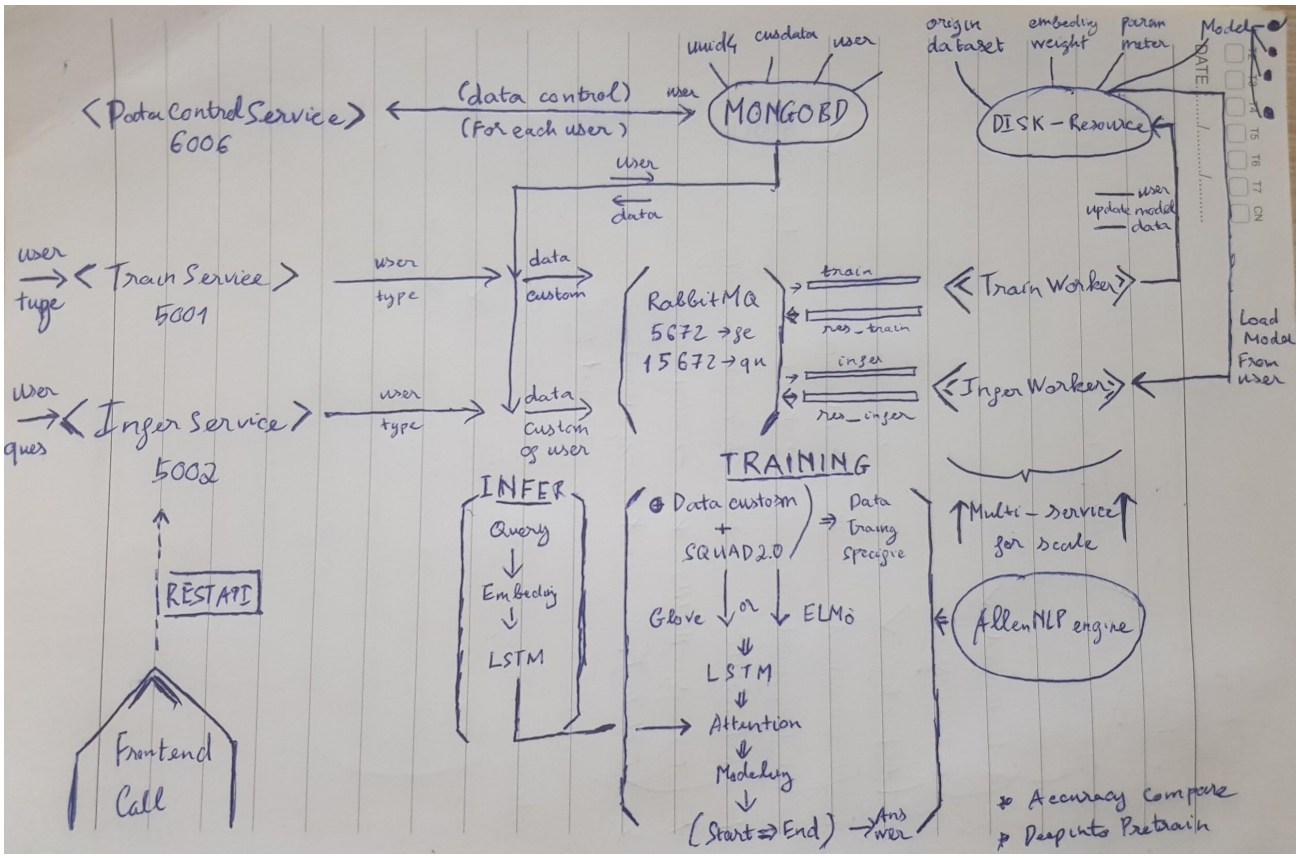
```
**Train from scratch with new dataset**
path: /train
Description: Train from scratch with current dataset
methods=['GET']
Required param: user

**Train with transfer-learning from lastest model**
path: /pretrain
Description: Train the lastest model, updated model
methods=['GET']
Required param: user
```

Infer_service: server/5002

```
**Inference from lastest model**
path: /infer
Description: Inference with given context and question, modelName is
[lastest] , [model1,2,3 ...], or default [origin]
methods=['GET']
Required param: user , question, [option] modelName
```

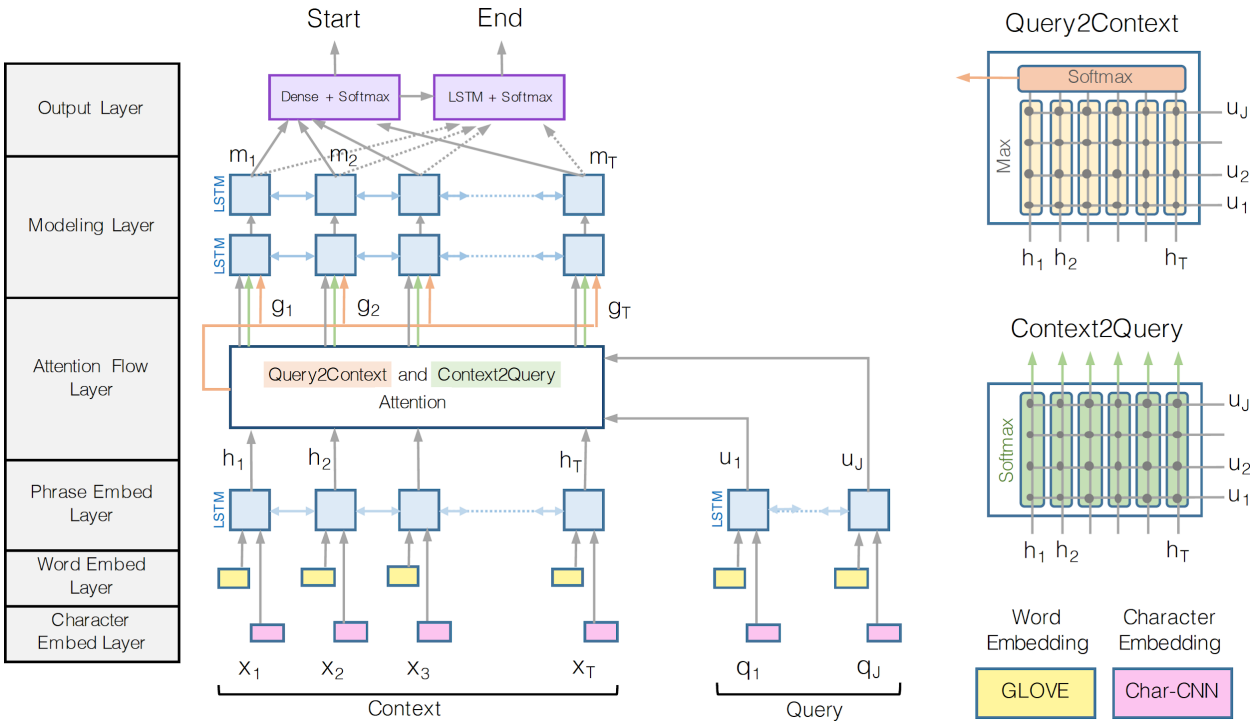
Architecture

Backend



AI models

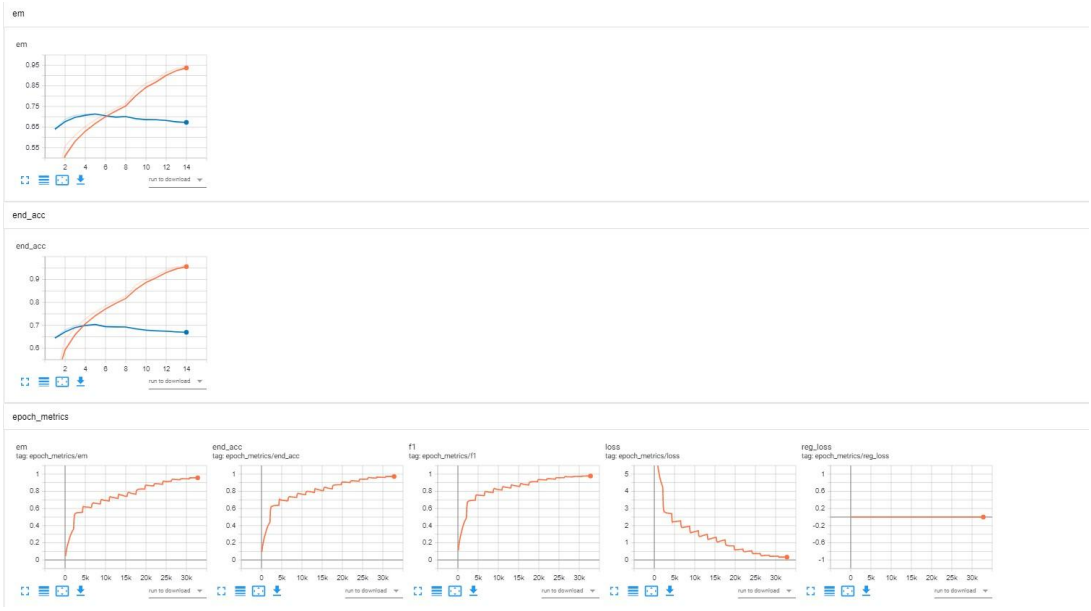
- BiDAF - **Bi-Directional Attention Flow:** https://arxiv.org/abs/1611.01603



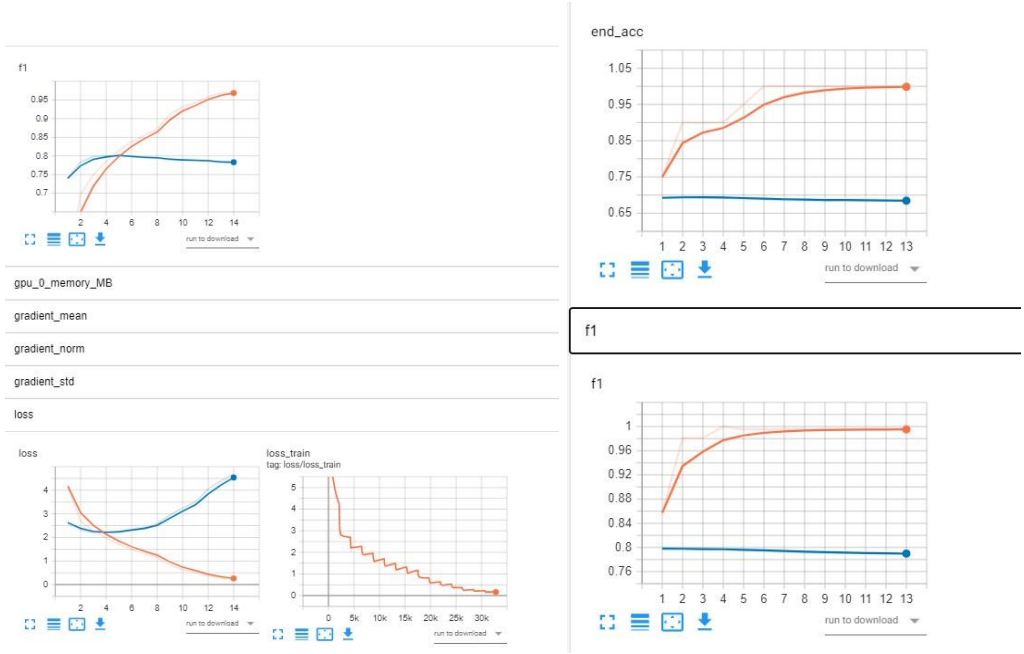
- **Statistic**

	Original Model	Train	Transfer learning
Dataset	SQUAD	SQUAD + custom	Custom data
Epoch		20	20
Training Time		1h 49min	34 min
Infer on general data	74.1%	70%	79%
Infer on specific data	42%	32%	99%

Training Process:



Merge Squad and new dataset for running on 1 GPU (AllenNLP 1.0.0 not support multi-GPU)



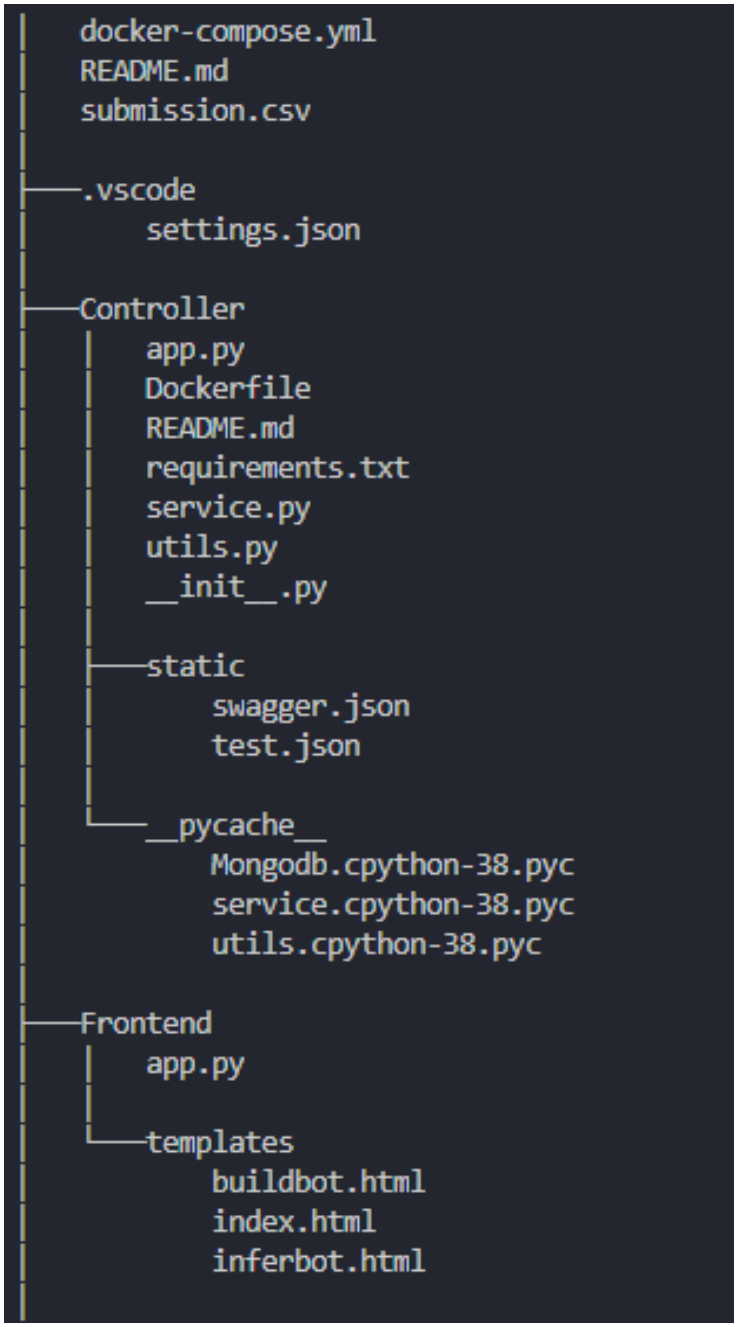
Merge full for training vs Pretrain on custom data

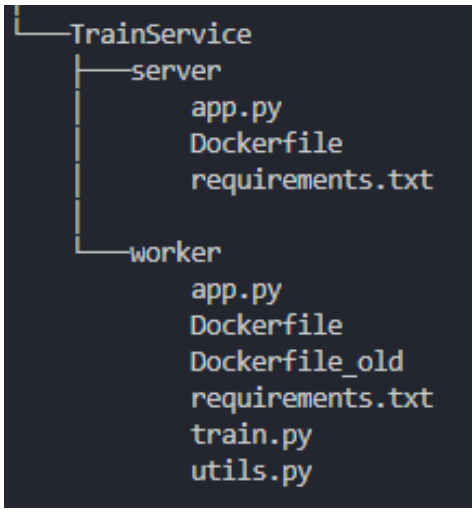
The loss validation (blue line) is tested on general dataset which is splitted from SQUAD:

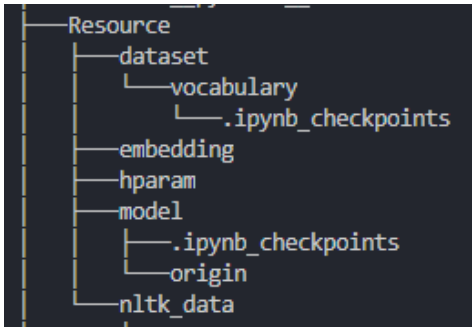
- If we use pretrain, reusing weight of the original model, when tested on a general dataset, the model is less **impacted**.
- When testing on specific data, Pretrain also perform better than Origin model
- One of drawback when using Pretrain is out-of-volcabulary problem

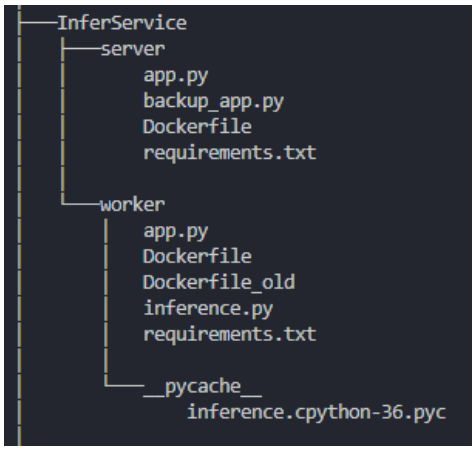
---

File Tree Diagram









Code highlight

AllenNLP custom for Pretrain:

General vs Specific

- If user data include many specific words, use train from scratch so that the new vocabulary can cover question for user context
- If vocabulary in dataset is general, use Pretrain for quick result

Pretrain problem - Model custom

```python
class Predictor_Pretrain(Predictor):
    @classmethod
    def from_archive_pretrain(cls,archive_path: str) -> "Predictor":

        #Init default
        plugins.import_plugins()
        cuda_device = -1
        predictor_name=None
        dataset_reader_to_load="validation"
        frozen = True

        archive=load_archive(archive_path, cuda_device=cuda_device)

        config = archive.config.duplicate()

        if not predictor_name:
            model_type = "bidaf" #config.get("model").get("type")
            model_class, _ = Model.resolve_class_name(model_type)
            predictor_name = model_class.default_predictor
        predictor_class: Type[Predictor] = Predictor.by_name(  # type: ignore
            predictor_name
        ) if predictor_name is not None else cls

        if dataset_reader_to_load == "validation" and "validation_dataset_reader" in config:
            dataset_reader_params = config["validation_dataset_reader"]
        else:
            dataset_reader_params = config["dataset_reader"]
        dataset_reader = DatasetReader.from_params(dataset_reader_params)

        model = archive.model
        if frozen:
            model.eval()
        return predictor_class(model, dataset_reader)
```

Painful Vocabulary

Message Queue Handle:

- Producer

```
class Inference(object):

    def __init__(self):
        self.connection = pika.BlockingConnection(pika.ConnectionParameters(host='rabbitmq'))

        self.channel = self.connection.channel()

        result = self.channel.queue_declare(queue='res_infer_queue', exclusive=False)
        self.callback_queue = result.method.queue

        self.channel.basic_consume(
            queue=self.callback_queue,
            on_message_callback=self.on_response,
            auto_ack=True)

    def on_response(self, ch, method, props, body):
        if self.corr_id == props.correlation_id:
            self.response = body

    def call(self, infer_content):
        self.response = None
        self.corr_id = str(uuid.uuid4())
        self.channel.basic_publish(
            exchange='',
            routing_key='infer_queue',
            properties=pika.BasicProperties(
                reply_to=self.callback_queue,
                correlation_id=self.corr_id,
            ),
            body=str(infer_content))

        while self.response is None:
            self.connection.process_data_events()

        self.connection.close()

        return str(self.response)
```

```
infer = Inference()
print(" [x] Inference model: "+str(user))
answer = infer.call(mes)
```

- Consumer

```
connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='rabbitmq'))
channel = connection.channel()
channel.queue_declare(queue='infer_queue', durable=True)
```

```
def callback(ch, method, properties, body):
    print(" [x] Received %s" % body)
    mes = body.decode()

    result = ""
    try:
        knowledge = ""
        question, context, modelName = mes.split("#$%")
        data = json.loads(context)
        for paragraphs in data["data"]:
            for para in paragraphs["paragraphs"]:
                knowledge = knowledge + para["context"]+". "
        modelName = "Resource/model/"+modelName+"/model.tar.gz"
        result = infer(question, knowledge, modelName)
    except:
        result = ""

    ch.basic_publish('', routing_key=properties.reply_to,
                     properties=pika.BasicProperties(
                     correlation_id=properties.correlation_id),
                     body=result)
    ch.basic_ack(delivery_tag=method.delivery_tag)
    print(" [x] Done")


channel.basic_qos(prefetch_count=1)
channel.basic_consume(queue='infer_queue', on_message_callback=callback)
channel.start_consuming()
```

MongoDB - SQUAD data structure:

Deploy - Docker-compose:

```yaml
version: "3"
services:

  mongodb:
    image: mongo:latest
    ports:
      - 27017:27017

  controller:
    build: Controller
    ports:
      - 6006:6006
    links:
      - mongodb
    depends_on:
      - mongodb

  rabbitmq:
    image: "rabbitmq:3.6-management-alpine"
    ports:
      - "5672:5672"
      - "15672:15672"
```

```yaml
  server_train:
    build: TrainService/server
    volumes:
      - ./TrainService/server:/app
    ports:
      - 5001:5001
    links:
      - mongodb
    depends_on:
      - mongodb

  server_infer:
    build: InferService/server
    volumes:
      - ./InferService/server:/app
    ports:
      - 5002:5002
    links:
      - mongodb
    depends_on:
      - mongodb
```

- Docker-compose and Docker run with GPU problem