11791 – hw5
Andrew id: xiaodit Name: Xiaodi Tao

Part 1

Implement the `Splitter` services. Currently the `Ranker` module (in particular the code the calculates similarity scores) performs sentence splitting and tokenizing every time two sentences are compared.

What I did:

1. In start.sh, I added Splitter in the loop of running service.py

```
#!/usr/bin/env bash

for dir in Splitter Expander Ranker Tiler Results; do
    echo "Starting $dir"
    cd $dir
    python service.py &
    echo "ending $dir"
    cd -
done
echo "Services started"
```

2. In pipeline.py, I adjusted the sequence of running each part

```
    bus.publish("splitter", message)
    count = count + 1
```

```
pipeline = ['expand.none','mmr.core', 'tiler.concat', 'results']
```

3. For service.py of Splitter, I added the main method

```
if __name__ == '__main__':
    print 'Starting Splitter'
    task = Splitter()
    task.start()
    print 'Waiting for the task to terminate.'
    task.wait_for()
    print 'Done.'
```

4. Inside Ranker, BiRanker, I removed the function of tokenizing and deleted them, because cu rrently the splitter will do the tokenizing.

5. In similarityJaccard.py, I implemented the calculateSimilarity method in order to calculate similarity.

```
class SimilarityJaccard(object):
    def __init__(self):
        self.stopWords = set(stopwords.words('english'))

    def calculateSimilarity(self, s1, s2):
        set1 = set([i.lower() for i in s1 if i.lower() not in self.stopWords])
        set2 = set([i.lower() for i in s2 if i.lower() not in self.stopWords])
        return float(len(set1.intersection(set2))) / len(set1.union(set2))
```

6. In CoreMMR.py, I change the objects of similarity calculation from sentence to sentence.tokens

```
    max_sent_sim = -99999999
    for other in best:
        # similarityInstance = SimilarityJaccard(sentence, other)
        sim = similarity.calculateSimilarity(sentence.tokens, other.tokens)
        if self.beta != 0:
```

Part 2

All of the services assume that the RabbitMQ server is available on *localhost*. In practice this is likely not to be the case. The address of the RabbitMQ server should be parameterized and obtained from an *.ini* file or loaded from an environmental variable.

What I did: I made the host to be a global variable and use it when calling MessageBus.

```
from dells.model import Serializer, DataSet, Question
host = os.environ.get('RABBITMQ_HOST')

if __name__ == '__main__':
    if len(sys.argv) == 1:
        print 'Usage: python pipeline.py <data.json>'
        exit(1)
    filename = sys.argv[1]
    print 'Processing ' + filename
    fp = open(filename, 'r')
    dataset = Serializer.parse(fp, DataSet)
    fp.close()


    pipeline = ['expand.none','mmr.core', 'tiler.concat', 'results']
    count=0
    bus = MessageBus(host=host)
    for index in range(0,10):
        question = dataset.questions[index]

        message = Message(body=question, route=pipeline)
        bus.publish("splitter", message)
        count = count + 1

    print 'Sent {} questions for ranking.'.format(count)
    print 'Done.'
```

Part 3

Deploy all of the services in a Docker container to simplify scaling and evaluation of your extended system. Ensure that the given scripts/modules (start.sh,pipeline.py,etc.) continue to work as originally specified in the README file (we will use these calls to test your container).

I deployed the docker image and pushed it to my Docker Hub as bioasq-xiaodit/FINAL. My Docker hub user name is xiaodit, password is Wangliyun1970.