COM-35702 Design and Engineering of Intelligent Information Systems

Engineering and Error Analysis with UIMA

Important dates

- **Hand out: April 4.**[a]
- **Turn in: April 25.**

[a]This version was built on April 4, 2014

Several notes about organizing your Maven project and other additional information:

1. **Submission:** Use the same process as for Homework 0, 1 and 2 (set up GitHub repo, create Maven project, write your code, submit to Maven repo), except that the name has changed to: hw4-ID.
2. **Your report for design:** We expect you to document how you designed the execution and deployment architecture for the sample information system . We will extract your documents from your submitted jar files. Remember to include your ID as part of the report file name, and put your name and ID inside your document. Please submit in PDF format only.
3. **Javadocs:** Please remember to give an appropriate description for each annotator you create.
4. Please post your questions regarding Homework 3 on Piazza `https://piazza.com/itam.mx/spring2014/com35702`. For other issues or concenrs, you can also send us mails to, Elmer Garduno (`elmer.garduno@itam.mx`).

Task 0.1 Creating Maven project from the archetype

For this task, we have prepared another archetype to help you quickly build your project. The tutorial for Homework 1 might help you create a Maven project from an archetype.

For Homework 4, you need to add the following Catalog URL

http://oaqa.github.io/DEIIS-hw4-archetype/repository/archetype-catalog.xml

The archetype for Homework is

hw4-archetype

Also remember that the **Group Id** and **Artifact Id** for Homework 4 are

edu.cmu.lti.11791.f13.hw4

and

hw4-ID

with ID being your Andrew Id.

Similarly, you need to edit the `pom.xml` file to type in the SCM information of your GitHub repository. You can see that different from Homework 1, we have also included the type system file with Java classes generated from `JCasGen` that your annotators might need. Your implementation starts here.

Task 1

Building Vector space Retrieval Model using UIMA

The purpose of this assignment is to design a simple vector space retrieval system using the UIMA framework. The vector space model is a widely used search engine model that represents query and document as vector of terms and uses the similarity between these vectors as a model of relevance for document ranking. We will use the cosine similarity measure for this task. Given that the underlying tasks are related to the work you did in homework2, you may wish to use some of the modules you have already developed. However, note that the evaluation metric is different from what you might have used earlier.

You will be given set of short documents. Your system should compute the Mean Reciprocal Rank (MRR) metric for tracking retrieval performance. MRR is useful metric when your system is supposed to rank the correct document at first position. The mean reciprocal rank is an average of the reciprocal ranks of results for a sample of queries Q:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

An initial pipeline code and a small text collection is provided to you. The code pipeline should work without any compilation errors but it does not produce any relevant results or evaluation measurements. We have already given comments inside the code (with //TODO:) so that you will know what is required. You can either fill in those gaps in the existing code, or reorganize the code yourself.

Use the following steps as a guideline to complete the assignment.

Task 1.1 Input

The text collection for this homework is given in documents.txt file in data directory of project. For simplicity we have restricted document length to only one sentence.

So, you can assume that your document contains only one sentence and no need to support multiple sentences in document for this homework.

Format of the file is described below:

```
qid=1 rel=99 John loves Mary
qid=1 rel=0 John and Mary are friends
qid=1 rel=1 Mary is loved by John
qid=1 rel=0 Mary likes John
qid=2 rel=99 ...
qid=2 rel=1 ...
qid=2 rel=0 ...
```

It contains information about the query ids, the relevant values and the corresponding text string. The relevant values of 1 indicates correct retrieval and relevant value of 0 denotes wrong retrieval results. The relevant value of 99 denotes the query itself. For example, "qid=1 rel=99 John loves Marry" means that the text string for qid 1 is "John loves Marry"

Task 1.2 Type System

The main program of the retrieval system is VectorSpaceRetrieval.java in the src directory with edu.cmu.lti.f13.hw4.hw4_ID package. This main program implements the aggregate analysis engine "VectorSpaceRetrieval.xml" on a text corpus. The descriptor for the engine is located in "descriptors/retrievalsystem/VectorSpaceRetrieval.xml".

The type system currently contains "Document", "Token" type. We have only added basic types and their features and you are supposed to make it richer depending on task requirements. At present, the type Document contains the following information: Relevance Value, Query ID, Text String, Token List. The Relevance Value, query ID, Text String are extracted from the text collection. The Token List represent 'bag-of-words' feature vectors and have to be constructed for the retrieval system. (Reminder: generate the TypeSystem implementation using UIMA JCasGen)

Hint: tokenList feature of Document needs to be populated by you using your annotators. Token type contains term and its corresponding frequency. You may want to write appropriated code to fill term and corresponding frequency during analysis.

Task 1.3 Analysis Engine

The aggregate analysis engine is composed of three primitive analysis engines, i.e., (1) DocumentReader, (2) DocumentVectorAnnotator, and (3) RetrievalEvaluator. The DocumentReader and the DocumentVectorAnnotator operate on individual document, but the RetrievalEvaluator operates on all documents. Detail description of the primitive analysis engines are available in "descriptor/retrievalsystem/DocumentReader.xml", "descriptor/engines/DocumentParser.xml", and "descriptors/retrievalsystem/RetrievalEvaluator.xml".

edu.cmu.lti.f13.hw4.hw4_ID.collectionreaders.DocumentReader is the first primitive analysis engine in the aggregate analysis engine as collection reader. It is going to

extract the relevant value, query id number, and sentence text from the text collection. This information information is kept in a CAS for further processing. DocumentReader descriptor is available at descriptor/retrievalsystem/DocumentReader.xml

We have written initial code for reading input data file in DocumentReader.java. You are free to change it on your own way, if require.

edu.cmu.lti.f13.hw4.hw4_ID.annotators.DocumentVectorAnnotator.java is a class that contains method 'createTermFreqVector(...)'. You need to implement the necessary code to update the tokenList of Document and update the CAS. DocumentVectorAnnotator analysis engine uses the CAS provided by DocumentReader. It need to extract the bag of word feature vectors from the text sentences. Specifically, the term and term frequency are filled with the word and word occurrence for that specific sentence. Its descriptor is available at descriptors/retrievalsystem/DocumentVectorAnnotator.xml

edu.cmu.lti.f13.hw4.hw4_ID.casconsumer.RetrievalEvaluator.java is a CAS consumer class where you would write code for finding cosine similarity between query sentence and each of the subsequent document sentence. The RetrievalEvaluator computes the Mean Reciprocal Rank (MRR) metric for the retrieval system. The MRR is averaged with respect to all sentences in the collection. The descriptor for RetrievalEvaluator is available at descriptor/retrievalsystem/RetrievalEvaluator.xml

[Hint: Inline java comments are given to implement the required pieces]

You will have to rank each of the potential document sentence by similarity score. Then, you will implement MRR metric using formula given earlier. Finally, you will have to find the MRR for all queries and print them on console. Your output should look like:

```
Score: 0.33806170189140655 rank=1 rel=1 qid=1 sent1
Score: 0.47140452079103173 rank=1 rel=1 qid=2 sent2
Score: 0.4472135954999579 rank=2 rel=1 qid=3 sent3
(MRR) Mean Reciprocal Rank ::0.8333333333333333
[The scores shown here are just examples]
```

Task 2

Error Analysis

Analyze and improve your retrieval system's performance by doing error analysis. Write briefly about why certain cases failed i.e. why the system could not give the highest rank to the correct document using the cosine similarity measure, what alternative strategy you adopted to resolve this problem, and why Once you test that everything works smoothly as expected, you need to summarize how you designed and implemented the type system for the task in your report. There is no template for the homework reports, but we expect you to consider the system design from several architectural aspects (UML, type system, engineering, design pattern, etc.) and several algorithmic aspects (knowledge sources, NLP tools, machine learning methods, etc.), documenting what you feel is the best report to communicate the essence of your design. E.g., novel design patterns are worth drawing a UML diagram for, but if you're using a well-known design pattern (e.g. Template Method) it's fine to document this with a brief textual description.

In summary, your tasks are:

1. automatically generate the type system implementation using UIMA¿¿JCasGen
2. correctly extract bag of words feature vector from the input text collection
3. compute the cosine similarity between two sentences in the text collection
4. compute the Mean Reciprocal Rank (metric) for all sentences in the text collection
5. Design a better retrieval system that improves the MRR performance measure
6. Improve the efficiency of the program by doing error analysis of retrieval system

We expect you to highlight the features of your design and your system. Finally, don't forget to put your name and Andrew ID at the top of the document, name the file as "hw4-ID-report.pdf" and put it under src/main/resources/docs. Describe your error analysis in detail, and include the relevant comparisons. The report is an important part of your submission, so be sure to allocate sufficient time to do a thorough job; feel free to consult the TAs if you have questions regarding the report.

Task 2.1 Bonus

Try other similarity measures instead of cosine similarity for ranking (i.e. dice coefficient, Jaccard coefficient) and include a comparison in your report.

HINTS

There is no preferred setup for this assignment, so you may write a new annotator to accomplish this task, thereby demonstrating your understanding of the UIMA framework and Java programming.

1. You are allowed to modify the provided type system
2. You are allowed to modify the provided classes and annotators. You are also allowed to create new methods, a new class, or even a new package.
3. If you can think of efficient design patterns for comparing multiple document ranking strategies in a single execution of the pipeline, feel free to implement!