OBJECTIVE

The goal of this assignment is to design a simple vector space retrieval system using the UIMA framework. Useful information on a vector space model can be found in the http://en.wikipedia.org/wiki/Vector_space_model. For a given text collection, the system is going to compute the Mean Reciprocal Rank (MRR) metric for the retrieval system based on the cosine similarity measure. Relevant Wikipedia articles are available at http://en.wikipedia.org/wiki/Cosine_similarity and http://en.wikipedia.org/wiki/Mean_reciprocal_rank

OVERVIEW

A template code and a small text collection will be provided to you. The code template does not has compilation errors but it is incomplete. Use the following steps as a guideline to complete the assignment.

1. The text collection for this assignment is available in the “dat” directory. 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 9 denotes the query itself. For example, "rel=9 qid=1 The shortest distance between new friends is a smile" means that the text string for qid 1 is “The shortest distance between new friends is a smile”
2. The main program of the vector space retrieval system is called VectorSpaceRetrieval.java under the src directory. This main program implements the aggregate analysis engine “VectorSpaceRetrieval.xml” on a text corpus. The descriptor for the engine is located in “descriptor/engines/VectorSpaceRetrieval.xml”.
3. The aggregate analysis engine is composed of three primitive analysis engines, i.e., (1) SentenceSplitter, (2) SentenceParser, and (3) SentenceEvaluator. The SentenceSplitter and the SentenceParser operate on individual sentence, but the SentenceEvaluator operates on all sentences. Detail description of the primitive analysis engines are available in “descriptor/engines/SentenceSplitter.xml”,“descriptor/engines/SentenceParser.xml”, and “descriptor/engines/SentenceEvaluator.xml”.
4. The type system for the analysis engine is available at TokenTypes.xml (available at descriptor/type_systems/TokenTypes). The type system TokenTypes contains the following information: RelevantValue, QueryID, TextString, BowWord, and BowFrequency. The RelevantValue, queryID, TextString are extracted from the text collection. The BowFrequency and BowWord represent 'bag-of-words' feature vectors and have to be constructed for the retrieval system. (**WARMUP: generate the TypeSystem implementation using UIMA>JCasGen**)
5. SentenceParser analysis engine uses the CAS provided by SentenceSplitter. It need to extract the bag of word feature vectors from the text sentences. Specifically, the bow_word and bow_frequency are filled with the word and word occurrence for that specific sentence. SentenceParser class and descriptor are available at src/retrieval/vector_space/engines/SentenceParser.java and descriptor/engines/SentenceParser.xml **(TODO: make sure the words and frequency of the bag-of-word feature vectors are correctly constructed.)**
6. The SentenceEvaluator computes the Mean Reciprocal Rank (MRR) metric for the retrieval system. The cosine similarity should be used to rank the sentences in the collection. The MRR is averaged with respect to all sentences in the collection. SentenceEvaluator class and descriptor are available at src/retrieval/vector_space/engines/SentenceEvaluator.java and

descriptor/engines/SentenceEvaluator.xml **(TODO: construct a bag-of-word feature vector for each sentence in the text collection. For each query, rank the retrieved sentences based on the cosine similarity measure. Compute the MRR for all query in the text collection.)**

7. SentenceSplitter is the first primitive analysis engine in the aggregate analysis engine SentenceAnalysis. 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. SentenceSplitter class and descriptor are available at src/retrieval/vector_space/engines/SentenceSplitter.java and descriptor/engines/SentenceSplitter.xml **(TODO: make sure information from the text collection are correctly extracted.)**

In summary, your tasks are:
0. automatically generate the type system implementation using UIMA>>JCasGen
1. correctly extract bag of words feature vector from the input text collection
2. compute the cosine similarity between two sentences in the text collection
3. compute the Mean Reciprocal Rank (metric) for all sentences in the text collection
4. (optional) Design a better retrieval system that improves the MRR performance measure
5. (optional) Improve the efficiency of the program and reduce the runtime of the retrieval system

REFERENCES

1. Download the UIMA Java framework & SDK version 2.4.0 at http://mirrors.sonic.net/apache//uima//uimaj-2.4.0/uimaj-2.4.0-bin.zip
2. Consult UIMA documentation at http://uima.apache.org/d/uimaj-2.4.0/index.html
3. Setup UIMA_HOME environment variable such as "C:\cr14\ja_bi\B12_uima\v240b\apache-uima"
4. Understand the concise introduction to UIMA, which is available at https://www.ibm.com/developerworks/webservices/tutorials/ws-uima.

HINTS

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

1. You are encouraged to implement the MRR metric yourself rather than using existing software package
2. You are allowed to modify the provided classes and annotators. You are also allowed to create a new methods, a new class, or even a new package.