Simranjit Singh Kohli
skohli@andrew.cmu.edu

Homework 4

# Task 1
# Building Vector space Retrieval Model using UIMA

## Task 1.1 Input

The sample data document consists of
Qid Rel DocumentData
Using regular expressions I seperated the data i.e. document Statement(Good friends long last) and metadata(Qid and releveance). An attribute followed by "\t" is the appropriate format.
Note: The acceptable format is
qid=1  rel=99 some text here
qid and **|tab|**rel=an|tab  textdata
an int val  int val  String

## Task 1.2 Type System

- The type system given in this homework is sufficient for the way retrieval is happening.
- Ideally for such a task we would require richer types.
- But in the current pipeline the CAS is bring reset and we are getting a stream of statements rather than a comprehensive document.
- Hence instead of focusing on a UIMA type system I focused on global structure for efficient storage retrieval.
- The system focuses on information processing hence a data centric architecture approach has been used for designing the system. A repository architectural pattern has been considered while desigining the types.
- With this in mind QueryGroupDictionary  (explained below) is implemented as a singleton.Becuase we need to creata a single memory to be implemented.
- Also a utility stopword Ontology has been created which stores stop word and checks for it.

**edu.cmu.lti.f13.hw4.hw4_skohli.interimtypes**

*FrequencyVector*
This type stores the a corresponding word and its frequency in the statement.
It is implemented by a map in which <Key,Value> is <word,frequency>.
The actual type used is a HashMap (preffered over HashTable since its unsynchronized and provides faster access).
Even though it is a map it is exposed via a class to define richer methods and utilities for retrieval and insertion(eg like filteredInsertion which checks for stop words and special char and doesn't insert them ).

*PersistantDocument*

Simranjit Singh Kohli
skohli@andrew.cmu.edu



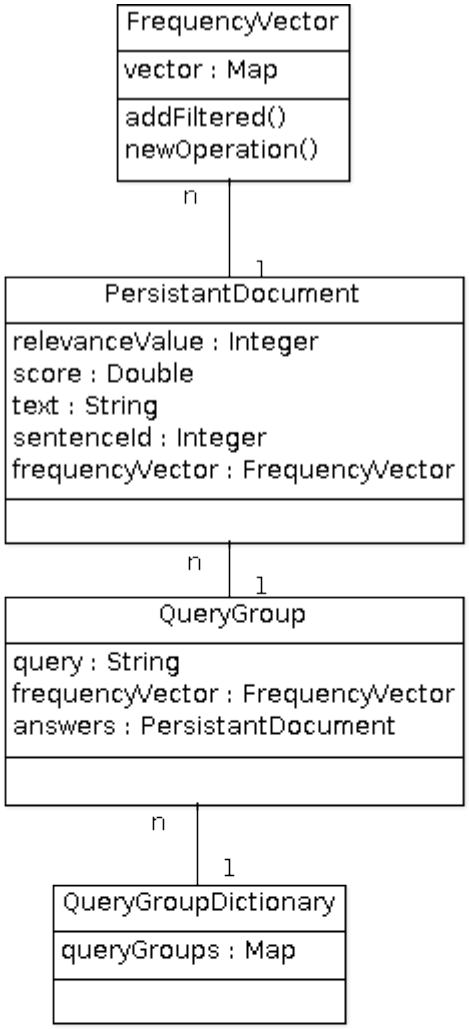
This type represents the document and metadata associated with it.
We need it since CAS is being reset.
**It has the following fields:-**
relevanceValue 1 for the right document and relevance value of 0 for an incorrect type.
score the score assigned by the algorithm used
text the actual sentence text.
SentenceId The number at which the sentence came in.
Along with this it also stores FrequencyVector of the sentence.

*QueryGroup*
It consists of the query (String )and  a list of PersistentDocuments.
It has been created since it provides better abstraction and view of data i.e aquery and documents associated with it.

*QueryGroupDictionary*
It consists of Map of a queryID and its associated queryGroup.
We need to maintain a global structure so that we can efficiently look for query groups to add and fetch data.
It is implemented by a map in which <Key,Value> is <queryId,querGroup>.
The actual type used is a HashMap

Simranjit Singh Kohli
skohli@andrew.cmu.edu

# Homework 4



Simranjit Singh Kohli
skohli@andrew.cmu.edu

Homework 4

Class diagram

## Task 1.3 Analysis Engine

The aggregate analysis engine is composed of three primitive analysis engines, i.e.,
(1)DocumentReader Reads the document and sends the query data in stream form line by line. Hence it operate on individual document.
(2) DocumentVectorAnnotator Generates the vector for each query or document.
(3) RetrievalEvaluator The evaluator assigns score to each retrieved query and computes the MRR. Hence the Retrieval Evaluator operates on all documents.

# Task 2
# Error Analysis

Since we are doing string matching do assign final scores we have to consider following concerns which can result in errors :-

**Case Dependency**: Can result in false negatives. Hence, it is a good idea to do case insensitive comparisons. I am storing the frequency vector terms in lower case to make it case independent.

**StopWords**: Words like a an etc can cause a decrease in scores. Filtering out stopwords can help remove fillers improve performance and accuracy.

**Lemmatization**: Converting words to their base form using lemmatization helps to get words in their root form. This helps to get a bettter match. I used stanford NLP jar for lemmatization.
Eg Booth shot Lincoln
Booth shoots Lincoln.
Both sentence have the same meaning. But normal approach this would get a score about 0.666 but using lemmatization we can get a score of 1.

**SpecialChar** : Lemmatization generates special characters.Filtering out special chars like “,?!” also help improve accuracy and speed.

# BONUS

I have implemented all the bonus similarity measures required for the hw4.Below is my evaluation of the similarity measures.

Simranjit Singh Kohli
skohli@andrew.cmu.edu

# Homework 4

Cosine similarity

It considers presence and frequency of words hence is a good measure for assigning scores to document.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Jaccard Index

I consideres only similarities at a boolean level i.e.
Index=common elements/total.
It does not consider frequency of words hence it is not a good measure of similarity.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Dice Coefficeint

Similar to Jaccard it only focuses on the different types of words present but doesnt take into account their frequency. Hence is not a good measure of similarity.

$$s = \frac{2|X \cap Y|}{|X| + |Y|}$$

This reasoning is backed by the data below with cosine function for similarity performing much better than Dice and Jaccard.

```
********************************************************************************
Calculating score using the Cosine Algorithm
Score:0.6123724356957945 rank=1 rel=1 qid=1 sent2
Score:0.5163977794943222 rank=2 rel=0 qid=1 sent3
Score:0.3849001794597505 rank=3 rel=0 qid=1 sent1

Score:0.4629100498862757 rank=1 rel=1 qid=2 sent3
Score:0.0 rank=2 rel=0 qid=2 sent1
Score:0.0 rank=3 rel=0 qid=2 sent2

Score:0.7071067811865475 rank=1 rel=0 qid=3 sent2
Score:0.5 rank=2 rel=1 qid=3 sent1
Score:0.5 rank=3 rel=0 qid=3 sent3

Score:0.3651483716701107 rank=1 rel=1 qid=4 sent2
Score:0.33806170189140655 rank=2 rel=0 qid=4 sent3
Score:0.31622776601683794 rank=3 rel=0 qid=4 sent1

Score:0.47140452079103173 rank=1 rel=1 qid=5 sent3
Score:0.3651483716701107 rank=2 rel=0 qid=5 sent1
Score:0.2886751345948129 rank=3 rel=0 qid=5 sent2
```

Simranjit Singh Kohli
skohli@andrew.cmu.edu

Homework 4

```
Cosine. (MRR) Mean Reciprocal Rank::0.8
Calculating score using the JacardIndex Algorithm
Score:0.3333333333333333 rank=1 rel=1 qid=1 sent2
Score:0.3333333333333333 rank=2 rel=0 qid=1 sent3
Score:0.125 rank=3 rel=0 qid=1 sent1

Score:0.3 rank=1 rel=1 qid=2 sent3
Score:0.0 rank=2 rel=0 qid=2 sent1
Score:0.0 rank=3 rel=0 qid=2 sent2

Score:0.5 rank=1 rel=0 qid=3 sent2
Score:0.3333333333333333 rank=2 rel=1 qid=3 sent1
Score:0.3333333333333333 rank=3 rel=0 qid=3 sent3

Score:0.25 rank=1 rel=0 qid=4 sent1
Score:0.2222222222222222 rank=2 rel=1 qid=4 sent2
Score:0.2 rank=3 rel=0 qid=4 sent3

Score:0.2857142857142857 rank=1 rel=1 qid=5 sent3
Score:0.2222222222222222 rank=2 rel=0 qid=5 sent1
Score:0.2222222222222222 rank=3 rel=0 qid=5 sent2

JacardIndex. (MRR) Mean Reciprocal Rank::0.6
Calculating score using the SorensonIndex Algorithm
Score:0.5 rank=1 rel=1 qid=1 sent2
Score:0.5 rank=2 rel=0 qid=1 sent3
Score:0.2222222222222222 rank=3 rel=0 qid=1 sent1

Score:0.46153846153846156 rank=1 rel=1 qid=2 sent3
Score:0.0 rank=2 rel=0 qid=2 sent1
Score:0.0 rank=3 rel=0 qid=2 sent2

Score:0.6666666666666666 rank=1 rel=0 qid=3 sent2
Score:0.5 rank=2 rel=1 qid=3 sent1
Score:0.5 rank=3 rel=0 qid=3 sent3

Score:0.4 rank=1 rel=0 qid=4 sent1
Score:0.36363636363636365 rank=2 rel=1 qid=4 sent2
Score:0.3333333333333333 rank=3 rel=0 qid=4 sent3

Score:0.4444444444444444 rank=1 rel=1 qid=5 sent3
Score:0.36363636363636365 rank=2 rel=0 qid=5 sent1
Score:0.36363636363636365 rank=3 rel=0 qid=5 sent2

SorensonIndex. (MRR) Mean Reciprocal Rank::0.6
Total time taken: 14.262
```