# 11791 - Homework 3
# Engineering and Error Analysis with UIMA

Mohammad Gowayyed
mgowayye@andrew.cmu.edu

October 22, 2014

**Abstract**

In this document, I report briefly my implementation for Task 1 of the homework, with MRR of **0.44**. I also perform a detailed error analysis for the various errors that occurred. I classify errors based on the their potential reason to: 1) errors due to non-lemmatization, 2) errors due to bad tokenization, and 3) errors due to being case sensitive in tokenization. I enhanced the system by using the Stanford Lemmatizer, developing a better tokenization method, that not just accounts for white spaces, but for other delimiters as well. The new system's MRR is **0.69**, with an increase of **56.82%**. To assess the statistical significance of the improvement, I used the t-test. The improvement is shown to be statistically significant with p-value of **0.003**, when treating ranks in both systems as matched pairs of observations.

## 1 Task 1

### 1.1 Overview

Check the Javadoc for a detailed explanation of the code. I followed the architecture provided. The only class I added is an inner class in the *RetrievalEvaluator* that is *Similarity*. *Similarity* stores the cosine similarity between a query and a document. It stores their indices in the ArrayLists that used to persist the data. It also stores the rank to be easily used while printing it to the results file.

**Data persistence** I used ArrayLists to store the documents data, the same as provided ArrayLists: *qIdList*, *relList*. I created additional two: *textsList* (that stores the original texts of the documents to be printed when needed) and *tokensList* that is an ArrayList of ArrayLists that store all the tokens with their frequencies.

## 2 Task 2

### 2.1 Error Analysis

In a spreadsheet file[1], I listed all the queries. For each query, I listed all the documents sorted by their cosine similarities. Then, I started to investigate[2] why the relevant document does not get high cosine similarity with the query.

Because the cosine similarity accounts for the *common tokens*, most of the errors happened because a token in the query is not mapped to itself in the document. Most of the mismatches are due to obvious reasons of either non-lemmatization, or bad tokenization. However, some mismatches are more complex like using an abbreviation as in New Jersey (in query 19) to N.J. (in the relevant document).

I believe that for huge data a better way would be to sample from the queries and investigate only the sample. The sample needs to cover some of the very bad results and some of the (near perfect) results, as well.

Table 1 shows a detailed information about why the mismatches occurs. Table 2 shows the counts of each mismatch reason. In the next subsection, I will show how I managed to improve the system to tackle these errors.

### 2.2 Improvements on the system

I applied three improvements on the system:

1. Improved the tokenization to tokenize over any non-alphabetic character.
2. Let the tokenizer ignore case
3. Used the provided Lemmatizer

It should be noted that these enhancements could make things worse in some cases (one case in our experiment - question 12), in the sense that it can lead an irrelevant document to get closer to the query, which was far because of the aforementioned problems. This problem can be solved by deriving a better similarity measure.

### 2.3 Statistical Significance Test

To show whether the improvements are statistically significant. I deal with the ranks as matched pairs of a random experiment. Then I perform a **t-test** and calculate the p-value, which is **0.003**. Which means that the ranking of the system after improvements is significantly less than the ranking of the initial system with degree of confidence **99.7%**. The calculations are in the attached spreadsheet. To calculate t-score, I used the function *ttest* in LibreOffice Calc.

[1]attached in the *doc* folder
[2]when applicable

| **Query ID** | **in query** | **in the relevant document** | **Suggested Solution** |
|---|---|---|---|
| 1 | volcanic | volcano | lemmatization |
| 2 | Jordan–one | Jordan | better tokenization |
| 3 | purchase | purchased | lemmatization |
| 4 | score | scored | lemmatization |
| 5 | China's | China | better tokenization |
| 5 | river | River | ignore case when tokenizing |
| 6 | minutes | minute | lemmatization |
| 6 | four minutes | four-minute | better tokenization |
| 7 | become | became | lemmatization |
| 7 | state? | state. | better tokenization |
| 8 | bite | bit | lemmatization |
| 9 | moon | Moon. | better tokenization and ignore case |
| 11 | Devil's | Devils | lemmatization |
| 12 | tallest | tall. | lemmatization and better tokenization |
| 13 | deep | depth. | needs dictionary and better tokenization |
| 15 | earth | Earth | better tokenization and ignore case |
| 16 | die | died | lemmatization |
| 17 | producer? | producer. | better tokenization |
| 17 | corn | Corn | better tokenization |
| 18 | McDonald's | McDonald | better tokenization |
| 19 | New Jersey | N.J. | this needs dictionary [3] |
| 20 | State? | State, | better tokenization |

Table 1: A detailed analysis of the reasons and suggested solutions for vocabulary mismatches.

| **Reason of vocabulary mismatch** | **Count** |
|---|---|
| bad tokenization | 11 |
| non-lematization | 9 |
| case sensitivity when tokenizing | 4 |
| use of abbreviations | 1 |
| others | 1 |

Table 2: Counts of the reasons of errors.

| **Lemmatization** | **Better Toknization** | **Ignore case** | **MRR** |
|---|---|---|---|
| | | | **0.44** |
| x | | | 0.55 |
| | x | | 0.55 |
| | | x | 0.46 |
| x | x | | 0.69 |
| x | | x | 0.55 |
| | x | x | 0.61 |
| x | x | x | **0.69** |

Table 3: Results of the different configurations of the improvements. Results are rounded to two decimal digits. We note that each enhancement alone increased the MRR, even without the others. We note also that Lemmatization and Better Tokenization had greater impact than Ignore case, individually and together, which is expected because our analysis shows that most errors can be solved using them.

| Initial system | After improvements |
|---|---|
| 2 | 2 |
| 2 | 1 |
| 3 | 1 |
| 2 | 1 |
| 3 | 2 |
| 2 | 2 |
| 3 | 2 |
| 2 | 2 |
| 2 | 1 |
| 1 | 1 |
| 4 | 1 |
| 3 | 4 |
| 3 | 3 |
| 2 | 1 |
| 3 | 3 |
| 3 | 1 |
| 3 | 2 |
| 2 | 2 |
| 3 | 3 |
| 2 | 1 |

Table 4: Ranking of relevant documents in the initial system and after improvements (with 0.69 MRR). We observe that the ranking of the improved system is always better, except in the question 12, because the enhancements in word matching that we provided actually made some irrelevant documents closer to the query even more than the relevant one.