

Information Search and Retrieval

LAB5 – Luke and
tuning Apache Lucene



The goal of this lab

- Better understand Lucene and in general IR systems
 - we will use Luke to explore index and our own implementations of custom parts of Lucene
 - we will design our own analyzer and see how it works
 - we will design our own scoring functions and see how it works
 - How does spelling work in Lucene?
 - How can we expand the query using synonyms?
 - Lucene benchmarking (evaluation)

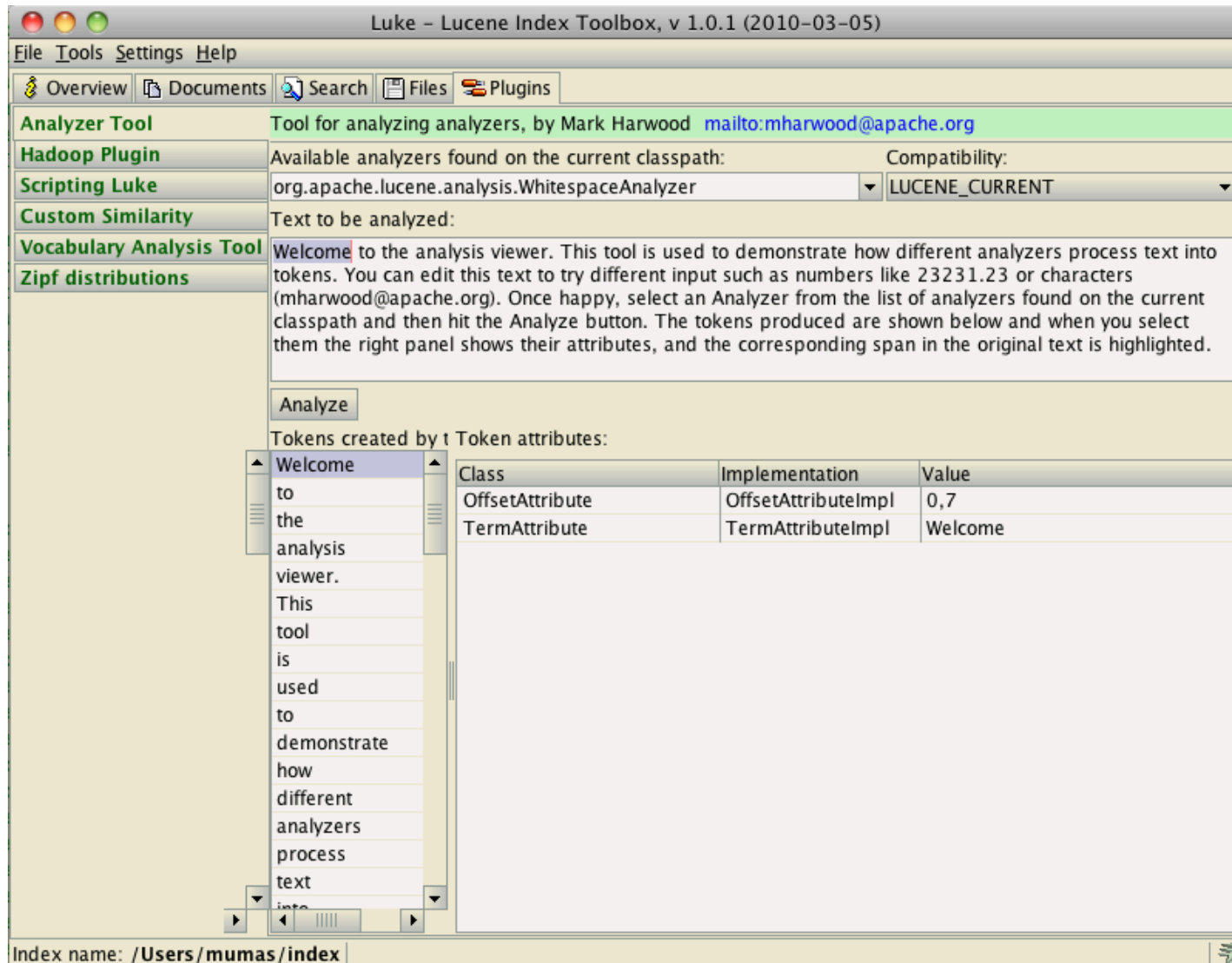
What do we need for this lab?

- ❑ Theoretical background (lectures)
- ❑ Understand what is Lucene
- ❑ Be able to run Lucene demo (indexing and searching)
- ❑ Create an index on lucene 3.0.1 java doc API

Obtaining Luke

- ❑ We will use not stable version of luke, because older does not work with lucene
- ❑ Checkout from svn and compile
 - svn checkout ***<http://luke.googlecode.com/svn/trunk/luke-read-only>***
 - build it using ant
 - you need ant and javac
- ❑ or download a jar bundle from here:
 - <https://www.inf.unibz.it/~lbaltrunas/Lab4/lukeall-1.0.1.jar>
- ❑ Run using java -jar lukeall-1.0.1.jar

Luke Demo



Investigate index

- ❑ Delete a document
- ❑ What is the next term in the dictionary for the filed:contents that comes after "apache"?
- ❑ Which documents contains these terms?

Analyzers

- ❑ Check how different analyzers work
- ❑ Implement your own simple analyzer and add it to the luke
 - remove common html tags from the indexed words
 - `java -classpath demo-analyzer.jar:lukeall-1.0.1.jar org.getopt.luke.Luke`
- ❑ Write 3-gram Analyzer and check how it works
- ❑ Make SnowBallAnalyzer work
 - what does it do differently comparing go Standard analyzer?

Scoring (recap.)

- ❑ make sure you read http://lucene.apache.org/java/3_0_1/scoring.html
- ❑ In Lucene, the objects we are scoring are Documents
- ❑ Lucene scoring works on **Fields** and then **combines** the results to return Documents
- ❑ Lucene allows influencing search results by "boosting" in more than one level
- ❑ Lucene combines Boolean model (BM) with Vector Space Model (VSM) of Information Retrieval
 - docs approved by BM are scored with VSM
 - http://lucene.apache.org/java/3_0_1/api/core/org/apache/lucene/search/Similarity.html

Changing default scoring

- ❑ You can change default scoring by rewriting
 - `org.apache.lucene.search.Similarity`
 - and setting it to `org.apache.lucene.search.Searcher.setSimilarity(Similarity similarity)`
- ❑ Implement your own similarity that does not take into account idf value
- ❑ Try to load it in Luke
 - search for a documents containing common term
 - check the explanations for the search

Benchmarking

- ❑ To run simple lucene benchmark:
 - get lucene source distribution
 - go to lucene-3.0.1/contrib
 - run ant run-task (reuters data set)
 - ant run-task -Dtask.alg=-Dtask.alg=conf/analyzer.alg
- ❑ Check available benchmarks in
 - lucene-3.0.1/contrib/benchmark/conf
- ❑ Read how to run your own benchmarks
 - http://lucene.apache.org/java/2_2_0/api/org/apache/lucene/benchmark/byTask/package-summary.html

Spelling

- ❑ Modify searching demo in such a way, that:
 - if 0 records returned because of a misspelled word
 - you would suggest rerun the search with the list of 5 words from the dictionary with the closest spelling
- ❑ You will need
 - <http://wiki.apache.org/lucene-java/SpellChecker>
 - lucene-3.0.1/contrib/spellchecker/lucene-spellchecker-3.0.1.jar

Query expansion using synonyms

- Find an easy way to expand Lucene query with synonyms
 - you do not need to implement
 - just find a package that can do it
 - check how to do it if you would need it