Lucene And Solr Document Classification

Alessandro Benedetti, Software Engineer, Sease Ltd.
Who I am

Alessandro Benedetti

- Search Consultant
- R&D Software Engineer
- Master in Computer Science
- Apache Lucene/Solr Enthusiast
- Semantic, NLP, Machine Learning Technologies passionate
- Beach Volleyball Player & Snowboarder
Agenda

- Classification
- Lucene Approach
- Solr Integration
- Demo
- Extensions
- Future Work
Classification

"Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known." Wikipedia
Real World Use Cases

- E-mail spam filter
- Document categorization
- Sexually explicit content detection
- Medical diagnosis
- E-commerce
- Language identification
Basics Of Text Classification

- Supervised learning
- Labelled training samples
- Documents modelled as feature vectors
- Term occurrences as features
- Model predicts unseen documents label
Apache Lucene

Apache Lucene™ is a high-performance, full-featured text search engine library written entirely in Java.

It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

Apache Lucene is an open source project available for free download.
Lucene index has complex data structures
Lot of organizations have already indexes in place
Pre existent data can be used to classify
No need to train a model from a separate training set
From training set to Inverted index
Apache Lucene For Classification

- Advanced configurable text analysis
- Term frequencies
- Term positions
- Document frequencies
- Norms
- Part of speech tags and custom payload
K Nearest Neighbours

- Given an index with labelled documents
- Each document has a class field
- Given an unknown document in input
- Given a set of relevant fields
- Search the top K most similar documents
- Fetch the classes from the retrieved documents
- Return most occurring class(es)
More Like This

- KNN uses Lucene More Like This
- Lucene query component
- Extract interesting terms* from the input document fields
- Build a Lucene query
- Run the query against the search index
- Resulting documents are “the similar documents”

* an interesting term is a term:
  - occurring frequently in the seed document (high term frequency)
  - but quite rare in the corpus (high inverted document frequency)
Assumptions
- Term occurrences are probabilistic independent features
- Terms positions are irrelevant (bag of words)

Calculate the probability score of each available class $C$
- Prior ($\frac{\text{#DocsInClassC}}{\text{#Docs}}$)
- Likelihood ($P(d|c) = P(t_1, t_2, ..., t_n|c) = P(t_1|c) * P(t_2|c) * ... * P(t_n|c)$)

Where given term $t$

$P(t|c) = \frac{\text{TF}(t) \text{ in documents of class } c + 1}{\text{#terms in all documents of class } c + \text{#docs of class } c}$

Assign top scoring class

Naive Bayes Classifier
Document Classification

- Documents are the Lucene unit of information
- Documents are a map field -> value
- Each field may be analysed differently (different tokenization and token filtering)
- Each field may have a different weight for the classification (affecting differently the similarity score)
Apache Solr

Solr is the popular, blazing fast, open source NoSQL search platform from the Apache Lucene project.

Its major features include powerful full-text search, hit highlighting, faceted search and analytics, rich document parsing, geospatial search, extensive REST APIs as well as parallel SQL.
Solr Integration

Index Time Integration - SOLR-7739
● Ingest the document
● Assign the class
● Set the class as a field value
● Index the document

Request Handler Integration (TO DO) - SOLR-7738
Return an assigned class :
● Given a text and a field
● Given an input document
● Given an indexed document id
Update Request Processor Chain

- Pipeline of processors
- Each single document flows through the chain
- Each processor is executed once
- Last processor triggers the update command
Update Request Processor

- Update Component
- Configurable Singleton Factory
- Single instance per request thread
- Process a single Document
- SolrCloud compatible*

* Pre processor / Post processor
Classification Update Request Processor

- Access the Index Reader
- A Lucene Document Classifier is instantiated
- A class is assigned by the classifier
- A new field is added to the original Document, with the class
- The document goes through the next processing steps
Solrconfig.xml - Update Handler

...<requestHandler name="/update" >
  <lst name="defaults">
    <str name="update.chain">classification</str>
  </lst>
</requestHandler>
...

Solrconfig.xml - Chain configuration

...  
<updateRequestProcessorChain name="classification">
  <processor class="solr.ClassificationUpdateProcessorFactory">
    ...
  </processor>
  <processor class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>

...
Solrconfig.xml - K nearest neighbour classifier config

```xml
<processor class="solr.ClassificationUpdateProcessorFactory">
  <str name="inputFields">title^1.5,content,author</str>
  <str name="classField">cat</str>
  <str name="algorithm">knn</str>
  <str name="knn.k">20</str>
  <str name="knn.minTf">1</str>
  <str name="knn.minDf">5</str>
</processor>

N.B. classField must be stored
Solrconfig.xml - Naive Bayes classifier config

```xml
<processor class="solr.ClassificationUpdateProcessorFactory">
    <str name="inputFields">title^1.5, content, author</str>
    <str name="classField">cat</str>
    <str name="algorithm">bayes</str>
</processor>

N.B. classField must be Indexed (take care of analysis)
Solr Classification - Important Notes

- Lucene >= 6.0
- Solr >= 6.1

- Classification needs a training set ->
  An index with initially human assigned classes is required
Solr Classification - Demo

- Sci-Fi StackExchange dataset
- Roughly 18,000 questions and answers
- 70% Training Set + 30% test set
Solr Classification - Demo

- Index the training set documents
  (this is our ground truth)
- Index the test set
  (classification will happen automatically at indexing time)
- Evaluate the test set
  (a simple java app to verify that the automatically assigned classes are consistent with what expected)
Solr Classification - Extensions SOLR-8871

Multi classes support
- Class field may be multi valued
- Assign multiple classes
- Not only the top scoring but top N (parameter)

Split human/auto assigned classes
- classTrainingField
- classOutputField

Default: use the same field
Classification Context Filtering

- Reduce the document space to consider -> reduce the training set
- Useful when only a subset of the index may be interesting for classification
- Consider only the human labelled documents as training data
Individual Field Weighting

- When classifying, each field has a different importance
  e.g.
  title vs content
- Set a different boost per field
- Knn compatible
- Bayes compatible
Solr Classification - Future Work

- Numeric Field Support (Knn) (Euclidean distance based)
- Lat lon support (Knn) (geo distance based)
- SolrCloud support (use the entire sharded index as training set)
Questions ?
• Special thanks to Tommaso Teofili, Apache committer who followed the developments and made possible the contributions.
• And to the Audience :)