

Multimedia Information Retrieval SDK

Part 2: Indexing, Searching, and Modelling

João Magalhães

Department of Computing

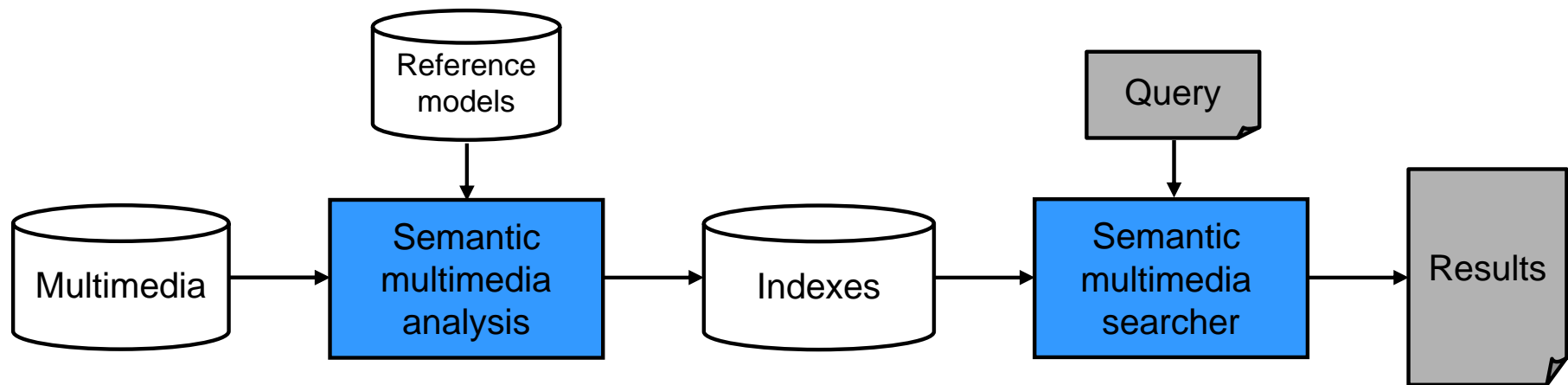
<http://www.doc.ic.ac.uk/~jmag/mir-sdk/>
j.magalhaes@imperial.ac.uk

Overview

- The multimedia information retrieval SDK is intended to help researchers and application developers.
- **Indexing**, **searching** and **modelling** tools enable multimedia IR applications
 - Similarity metrics for search by example applications. (Kulback-Leibler, Jensen-Shannon, Histogram-Intersection, Cosine, Minkowski)
 - Multimedia analysis for indexing applications.
 - Modelling algorithms to update existing models. (Rocchio, Naïve Bayes, LS, Logistic regression with L1 and L2 penalties)

Indexing and searching

- The indexing task invokes the semantic multimedia analyser with the specified reference models
- The search task invokes the semantic multimedia searcher with the metric specified in the query
- Both tasks use the settings of the reference models








Reference models

- Reference models are at the core of the toolkit
- In multimedia and semantic analysis one needs to create a model of the concept based on some examples
 - This contrasts with traditional text tokenizers that are hardcode
- Reference models contain:
 - A vocabulary of concepts that shall be used to index content
 - A model for each concept
 - Settings files for indexing, searching and modelling

Reference models

Reference Models

	indexes	Annotations of training media and corresponding vocabulary
	models	Mixture models and linear models are stored here.
	indexer.xml	Settings file to index new content
	searcher.xml	Settings file for search tool
	learner.xml	Settings file for learning models

Reference models: settings files

- There is a settings file for each task: indexing, search, and modelling

```
<?xml version="1.0"?>  
<MIRToolkit Application="indexing" MessagesLevel="quiet">
```

```
<!-- Dataset related settings -->  
<Datasets> . . . </Datasets>
```

```
<!-- Multimedia data representation settings -->  
<MultimediaDataRepresentation> . . . </MultimediaDataRepresentation>
```

```
<!-- Semantic multimedia analysis settings -->  
<MultimediaModels> . . . </MultimediaModels>
```

```
<!-- Semantic multimedia search settings -->  
<MultimediaSearch> . . . </MultimediaSearch>
```

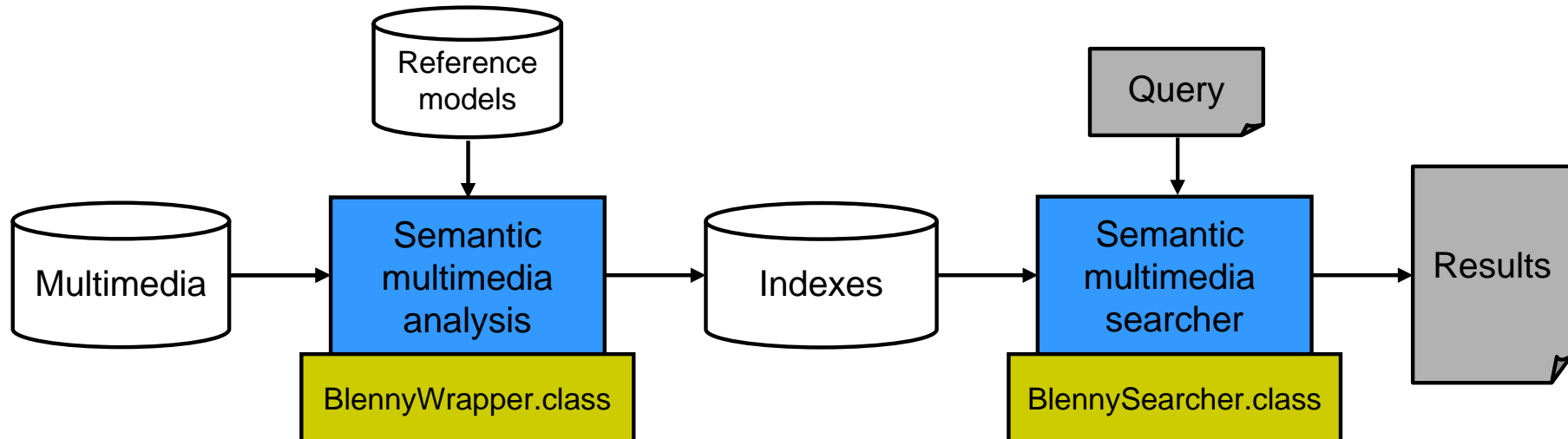
```
</MIRToolkit>
```

Included reference models

- Text dataset: reuters-21578
 - Models 90 categories of news articles
 - Features are a bag-of-words
 - Includes several types of models for each category
- Images dataset: Corel images
 - Models 100 visual concepts of photographic models
 - Features are based on colour and texture descriptors
 - Includes several types of models for each concept
- Multimodal dataset: TRECVID2005
 - Models 30 multimedia concepts of TV video content
 - Features are based on colour and texture descriptors, and a bag-of-words
 - Includes several types of models for each concept

Goby: the Java interface

- Goby provides interfaces for the basic tasks of toolkit

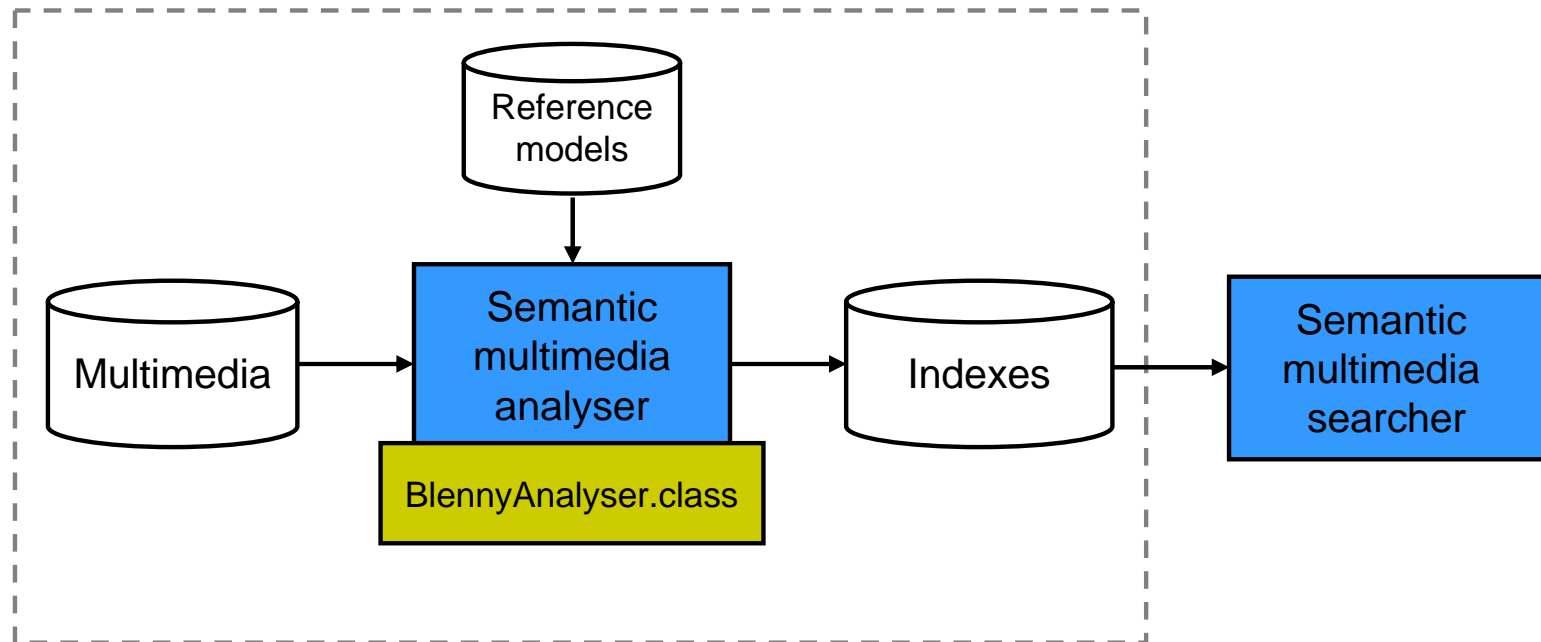


Java interface: BlennyWrapper.class

```
public class BlennyWrapper {  
  
    // Methods to access the dataset  
    public BlennyWrapper(String referenceModels, String newIndexes)  
    public Vector<String> getListConcepts()  
    public Vector<String> getListModels()  
  
    // Methods for information extraction  
    public boolean setModel(String model)  
    public boolean analyseMedia()  
    public Float getAnnotations(String mediaUri, String concept)  
  
    // Methods for search  
    public boolean setMetricSpace(String space)  
    public TreeMap<String, Float> searchMedia(MediaQuery query)  
  
    // Methods for learning models  
    public boolean learnAllConcepts()  
    public boolean learnConcept(String concept)  
  
}
```

Multimedia indexing

- The indexing application infers the concepts probabilities.
- The reference models that were built during the learning phase and follow a given vocabulary



Multimedia indexing example

```
// Creates a new analyser with reference models but with no new content
Analyser = new BlennyWrapper("C:\\RefDataset\\", "C:\\NewMedia\\");

// Get the vocabulary
Vector<String> vocabulary = Analyser.getListConcepts();

// Get a concept from the list of concepts
String concept = "people";

// Get the list of configured models
Vector<String> models = Analyser.getListModels();

// Select the model from the above list and set the current concept models
Analyser.setModel("NaiveBayes");

// Invokes the analyser on the specified directory
Analyser.analyseMedia();

// docUri must point to a valid segment of a document
String docUri = "C:\\NewMedia\\...";

// Get the probability of concept people on the given media
Float prob = Analyser.getAnnotations(docUri, concept);
```

Data preparation

- The path “C:\NewMedia\” must contain a file “index.txt” with metadata about the new documents
- Documents should be segmented into meaningful units to be analysed
 - These segment units are dependent of the application domain
 - Each segment can be <text>, <images>, or pairs <text, images>
- The segments analysis will produce concepts to index documents by their segments

Dataset metadata format

- The toolkit expects data in a text file where each segment has the following structure:

Line 1: DocumentId

Line 2: SegmentId

Line 3: ImageId

Line 4: Text content

Line 5: Link to visual content

Line 6: Positive annotations

Line 7: Negative annotations

Line 8: Null line

```
ExampleDocument123.odf
```

```
ExampleSegment123_ABC
```

```
ExampleImage123_ABC_12
```

```
here goes the text features
```

```
/somePath/ThisIsImage123_ABC_12.jpg
```

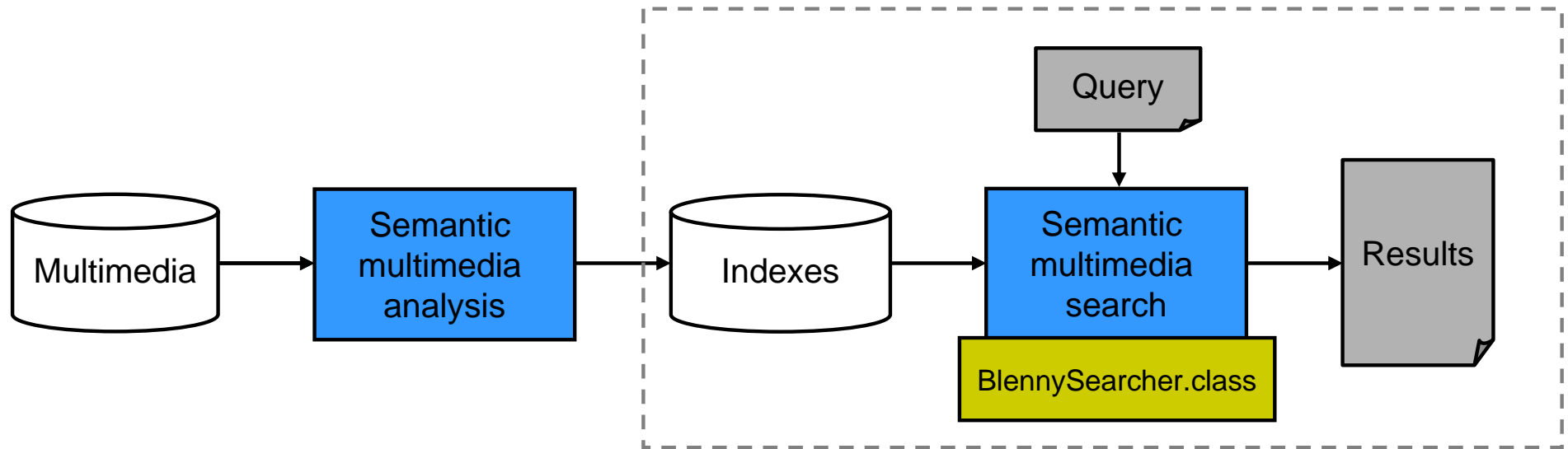
```
face person studio
```

```
airplane animal boat_ship building bus...
```

```
=====
```

Multimedia search

- The toolkit provides three types of search:
 - Search by keyword; Search by example; Search by semantic example
- The query is composed by a weighted logical combination that can mix semantic examples and keywords.



Multimedia searching example

```
// Creates a new analyser with reference models but with no new content  
Analyser = new BlennyWrapper("", "C:\\NewMedia\\");
```

```
MediaQuery query = new MediaQuery();
```

```
// Add a keyword to the query
```

```
query.addQueryUnit("people", 1.0f,  
                   MediaQuery.QueryType.KEYWORD,  
                   MediaQuery.QueryOperator.AND);
```

```
// Add a semantic example to the query
```

```
query.addQueryUnit("someUri", 1.0f,  
                   MediaQuery.QueryType.SEMANTIC_EXAMPLE,  
                   MediaQuery.QueryOperator.AND);
```

```
// Sets the search space (metrics and input)
```

```
Blenny.setMetricSpace("Cosine");  
Blenny.setModel("NaiveBayes");
```

```
// Runs search task and gets top 100 results
```

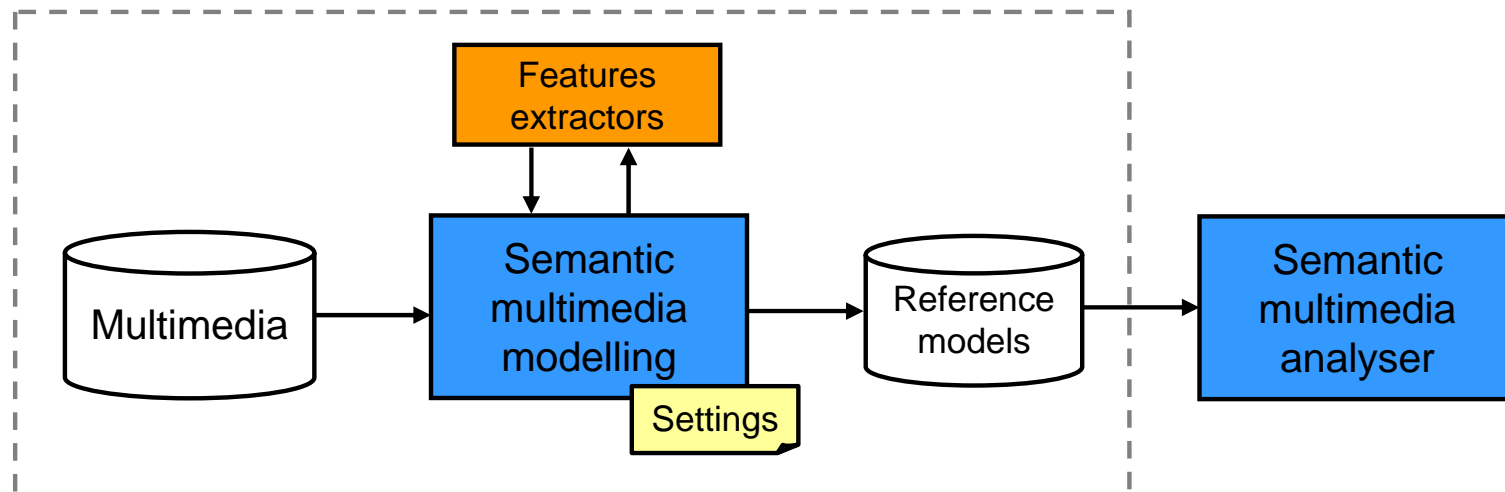
```
TreeMap<String, Float> Results = Blenny.searchMedia(query, 100);
```

Query construction

- A query is composed by several **QueryUnits**
- Query units processing is equivalent to **stack calculators**
- Query units are merged with **AND** or **OR** and have a specific weight
- Their type also specify the search space:
 - If by KEYWORD it has as many dimensions as keywords
 - If by EXAMPLE it has as many dimensions as the feature-space
 - This case is not implemented yet
 - If by SEMATIC_EXAMPLE it has as many dimensions as the semantic-space

Multimedia modelling

- The indexing application infers the concepts probabilities.
- The reference dataset define the ontology and the models built during the development phase



Multimedia modelling example

```
// New training data and new annotations are loaded from "C:\RefModels\index.txt"  
// Existing indexes and vocabulary will be extended to accommodate new concepts
```

```
// Creates a new analyser with reference models but with no new content  
Analyser = new BlennyWrapper("C:\\RefDataset\\", "");
```

```
// Sets the working model  
Blenny.setModel("NaiveBayes");
```

```
// Learns the models of all concepts (USE WISELY....)  
Blenny.learnAllConcepts();
```

```
// Learn the concept of a single concept  
Blenny.learnConcept("people");
```

Multimedia modelling: some notes

- Learning new models and adding new training data should be done carefully
- Models estimation has several parameters that have an expensive computation
 - Settings: consider limiting the amount of training data, the number of CV folds, and the granularity of the hyper-parameters
- Adding **new training data** to learn an existing concept might produce an inconsistent set of models
- Adding **new annotations** of a specific concept and learning it's new model does not existing models

Summary

- Indexing methods infer indexing tokens (keywords) with associated weights
 - This inference is done according to the available models
- Searching methods allow to search new media for arbitrary combinations of keywords and examples
- Modelling allows to:
 - Update models of concepts with new annotations and/or new training data
 - Add new concepts