Statistical Language Modeling for Information Access

Practical 2: Retrieval and evaluation

Maarten de Rijke Edgar Meij Krisztian Balog

University of Amsterdam
Norwegian University of Science and Technology

August 1–4, 2011

Outline of the Course

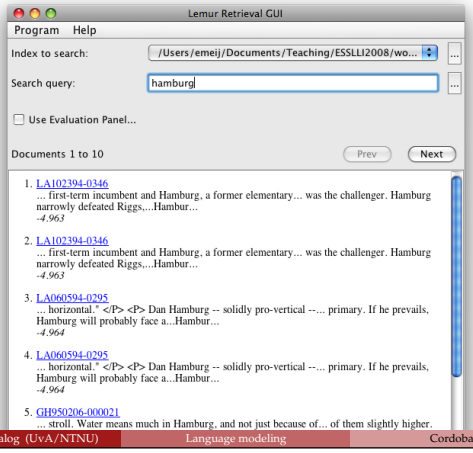
Practical

- Day 1: Installing and Indexing
- Day 2: Retrieval and Evaluation
- Day 3: Retrieval Parameters and Indri
- Day 4: Pseudo Relevance Feedback and Some More Evaluation; Additional bells, whistles and requests

Looking back

- Downloaded/Installed Lemur
- Downloaded the CLEF 2006 adhoc collection
- Created an IndriBuildIndex parameter file
- Ran the indexer
- Viewed the output using LemurRet.jar
- Questions?
- You can always e-mail me later: `emeij@science.uva.nl`

LemurRet.jar



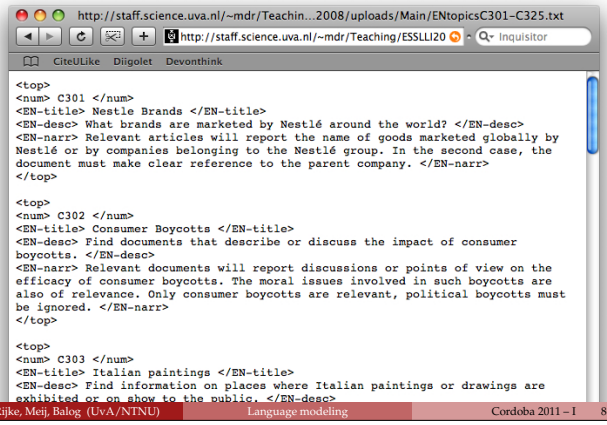
Outline

1. **Retrieval**
   - Queries
   - Retrieval
   - Retrieval Models
2. **Evaluation**
   - qrels
   - trec_eval
3. **Exercises**
   - Exercises

Retrieval

- General pipeline, given a set of queries:
  - Preprocess/Transform queries
  - Perform a retrieval run, using some settings
  - Evaluate
- Queries aren't usually in the "proper" format...

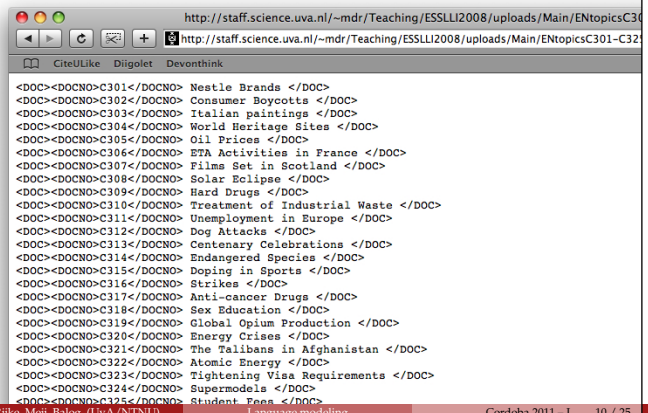
Queries — original format



Preprocessing

- Need to perform some kind of preprocessing to create TREC Text/Web format
- Use Perl/bash/awk/...
- Example topics and script on the wiki

Queries — TREC Web format



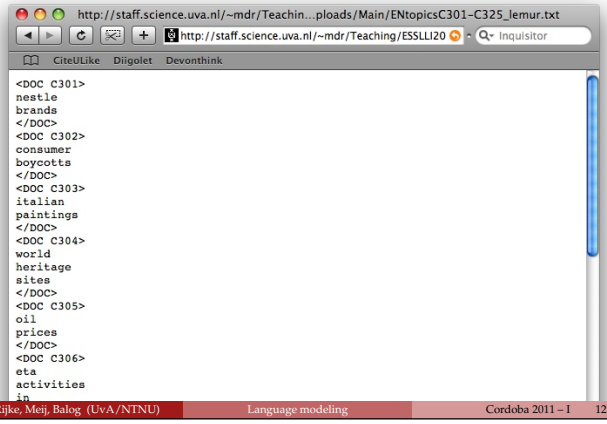
ParseToFile

- But then we're still not there yet
- Lemur only understands Basic Doc Format (LDF)
- Use ParseToFile to parse an inputfile containing queries (TREC Text/Web format) into LDF format
- Steps:
  - Create ParseToFile parameter file (similar to the indexer's config file), e.g.:

```
<parameters>
<docFormat>web</docFormat>
<outputFile>path/to/outputfile.ldf</outputFile>
</parameters>
```

  - Run: `ParseToFile [parse_param] [query_file]`

Queries — LDF format



Retrieval

- Works in the same way as indexing
  - Create parameter file
  - Basic parameters:
    - index - the index to use
    - retModel - the model to use
    - textQuery - the query file to use
    - resultCount - number of results
    - resultFile - where to store the output
    - **TRECResultFormat** - to use TREC-style output
  - run `RetEval [param_file]`

Example RetEval parameter file

```
<parameters>
<index>/path/to/your/index</index>
<retModel>kl</retModel>
<textQuery>path/to/queries.ldf</textQuery>
<resultCount>1000</resultCount>
<resultFile>queries.res</resultFile>
<TRECResultFormat>1</TRECResultFormat>
</parameters>
```

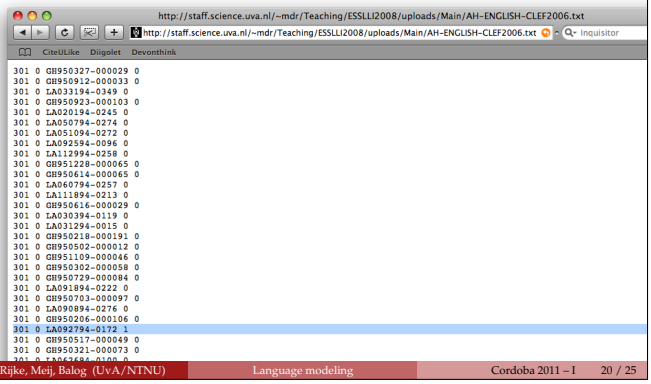

Retrieval Models

- Lemur supports a number of retrieval models:
  - **kl** — KL-divergence (query-likelihood), with
    - Jelinek-Mercer smoothing
    - Dirichlet smoothing
    - Absolute discount
    - Two-stage smoothing
  - **tfidf** — TF.IDF
  - **cos** — Cosine
  - **okapi** — Okapi (BM25)
- You can (easily) implement your own!
- More on these tomorrow…

Small demo

qrels

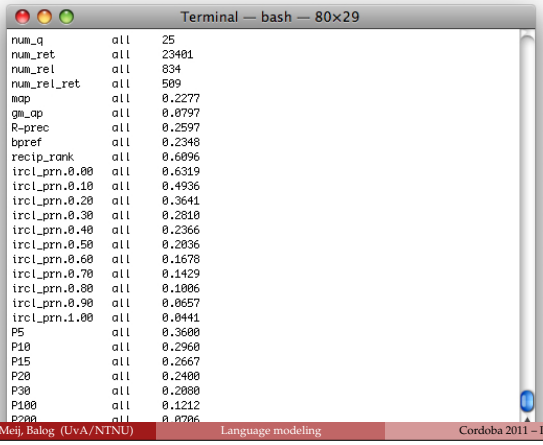
A "qrel-file" contains the judgments on a document collection and set of queries



trec_eval

- Written by Chris Buckley for the TREC evaluations
- Outputs evaluation measures
  - precision
  - recall
  - MRR
  - MAP
  - And many many more...
- Both for runs and individual topics (use the `-q` flag)
- Usage:
  `trec_eval [path/to/qrels/] [path/to/resultfile]`

output



Exercises

- Compare results of two or more different retrieval models on the CLEF collection
- Report on (interesting) differences
- Where does another model help? hurt? In terms of precision, recall or some average?
- Why?