Statistical Language Modeling for Information Access

Practical III: Retrieval Parameters

Maarten de Rijke Edgar Meij Krisztian Balog

University of Amsterdam
Norwegian University of Science and Technology

August 1–4, 2011

Outline of the Course

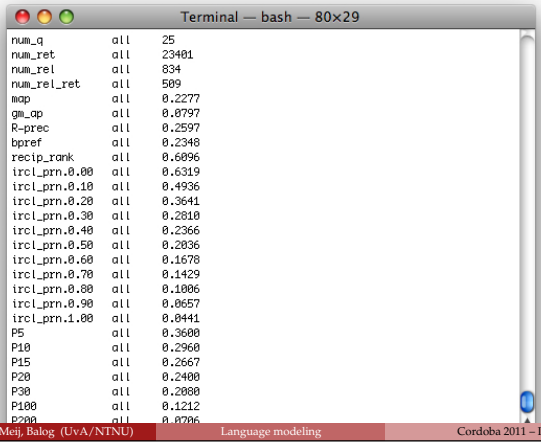
Practical

- 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

- Parsed the CLEF 2006 adhoc queries
- Created a RetEval parameter file
- Performed a retrieval run
- Evaluated the output using trec_eval
- Questions?
- You can always e-mail Edgar Meij at `edgar.meij@uva.nl`

trec_eval



Outline

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: TF.IDF with length normalization
  - **okapi** — Okapi/BM25
- You can (easily) implement your own or modify existing models

KL-Divergence

- Ranks documents according to the KL-divergence with the (empirical) query model
- *Negative* KL-divergence is rank-equivalent to query-likelihood (as seen yesterday)
- Lemur outputs KL-divergence values by default
- To change this, use the parameter `adjustedScoreMethod`
  - "querylikelihood" or "ql" for query likelihood
  - "crossentropy" or "ce" for cross entropy
  - "negativekld" or "-d" for negative KL divergence
- Note that this only changes the output scores!

Smoothing

- In order to apply document smoothing during retrieval, use the `smoothMethod` parameter
  - **jm** – Jelinek-Mercer smoothing
  - **dir** – Dirichlet smoothing
  - **ad** – Absolute discounting
  - **2s** – Two-stage smoothing
- Smoothing parameter values are also controlled by parameters in the config file:
  - **JelinekMercerLambda** – collection model weight λ (default 0.5)
  - **DirichletPrior** – prior parameter μ (default 1000)
  - **discountDelta** – discounting constant (default 0.7)

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>
<smoothMethod>jm</smoothMethod >
<JelinekMercerLambda>0.15</JelinekMercerLambda>
</parameters>
```


Indri

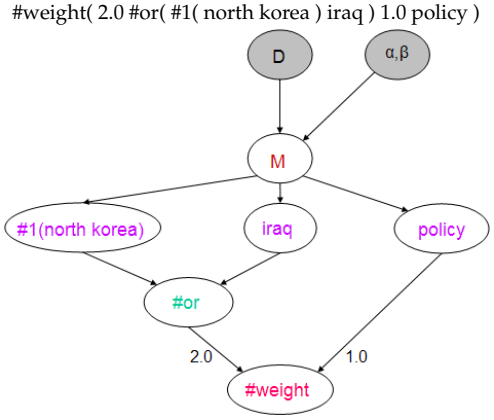
- Indri is the largest type of Lemur (on Madagascar)
  - "Indri" is Malagasy for "Look!"
- Most of the code in the Lemur toolkit is part of the Indri project, e.g. the indexing codebase from Lemur has now mostly been delegated to Indri (which can handle much larger collections of documents and understand the structure of HTML/XML documents)
- We've already used the `IndriBuildIndex` on day 1!
- Lemur is, unlike Indri, geared towards developing retrieval models. However, Indri incorporates the Indri query language which enables elaborate query formulation strategies

Indri Structured Query Language

- Based on InQuery
  - J.P. Callan, W.B. Croft, J. Broglio. TREC and Tipster Experiments with Inquery. In: *Information, Processing, and Management*, 1995
- InQuery still part of the Lemur toolkit
  - Use `ParseInQueryOp` instead of `ParseToFile`
  - Use `StructQueryEval` instead of `RetEval`
  - Use `<retModel> inquery </retModel>`
- InQuery is based on the notion of so-called *Belief networks* (aka Bayesian/inference networks)
- Indri uses the same ideas, but applies language modeling techniques for the estimation of beliefs

Belief Network

#weight(2.0 #or(#1(north korea) iraq) 1.0 policy)



Indri Structured Query Language

- Query term weighing
- Phrases / Windowed search
- (Weighed) Synonyms
- OR / NOT / MAX
- Passages / Filters
- *Nesting*
- (Fielded search)
- (Mixture models on document fields)
- (Document priors — also in Lemur)

Query Likelihood and Synonmys

```
#combine( ESSLLI Hamburg )
```

- Is interpreted by Indri as
  $score = \log(P(\text{"ESSLLI"}|M)) + \log(P(\text{"Hamburg"}|M))$
- $P(\cdot|M)$ is a smoothed estimate
- More on smoothing in Indri later

```
#syn( car automobile )
```

- Occurrences of "car" or "automobile"

Weights

```
#weight( 1.0 ESSLLI 0.5 Hamburg )
```

- Is interpreted by Indri as
  $0.67 \log(b(\text{"ESSLLI"})) + 0.33 \log(b(\text{"Hamburg"}))$, where $b(w) = P(w|M)$
- Can be used for query expansion/modeling
- More on this tomorrow

Phrases

```
#odn( White House )
```

- Ordered window
- "White" n terms **before** "House"

```
#udn( White House )
```

- Unordered window
- "White" **within** n terms of "House"

OR / NOT / MAX

```
#not( ESSLLI )
```

- Pages that do not contain ESSLLI
- *score* = $\log(1 - b(\text{“ESSLLI”}))$

```
#or( ESSLLI Hamburg )
```

- Pages that contain “ESSLLI” or “Hamburg”
- *score* = $\log(1 - (1 - b(\text{“ESSLLI”})) \cdot (1 - b(\text{“ESSLLI”})))$

```
#max( ESSLLI Hamburg )
```

- Returns maximum of b(“ESSLLI”) and b(“Hamburg”)

Passages and Filters

```
#combine[passage200:100]( ESSLLI )
```

- Create passages of length 200 every 100 words and rank each by $P(\text{“ESSLLI”}|M_{passage})$

```
#filreq( ESSLLI #combine ( Hamburg ) )
```

- Rank documents that contain “ESSLLI” by #combine (Hamburg)

```
#filrej( ESSLLI #combine ( Hamburg ) )
```

- Rank documents that **do not** contain “ESSLLI” by #combine (Hamburg)

Running Indri queries

- Again, a parameter file. . . slightly different syntax though
- There is just one retrieval model: KL-divergence
- Basic parameters:
  - **index** – the index to use
  - **rule** – the smoothing to use
  - **query** – the **query** to use (can be specified multiple times)
  - **count** – number of results
  - **trecFormat** – to use TREC-style output
- output goes to STDOUT

Smoothing

- Smoothing and parameters are defined by the “rule” field
  - **dir** – Dirichlet smoothing
  - **jm** – Jelinek-Mercer smoothing
  - **two** – Two-stage smoothing
- Examples:
  - `<rule>method:dir,mu:1000</rule>`
  - `<rule>method:jm,lambda:0.15</rule>`
  - `<rule>method:two,mu:1000,lambda:0.15</rule>`

Example

```
<parameters>
<index>/path/to/your/index</index>
<rule>method:jm,lambda:0.15</rule>
<count>1000</count>
<trecFormat>1</trecFormat>
</parameters>
```


Queries

- Different queries... different format

```
<parameters>
<query>
<number>1</number>
<text>this is the first query</text>
</query>
<query>
<number>2</number>
<text>this is another query</text>
</query>
</parameters>
```

- Indri script and query file are again on the wiki
  - `http://www.science.uva.nl/~mdr/Teaching/ESSLLI2008`
  - LostInHamburg (case sensitive!)

Running

- Good practice to separate the configuration from the queries
- Run `IndriRunQuery [param_file] [query_file] > outputfile`
- Contrary to Lemur, Indri can also handle command-line switches
- `IndriRunQuery [param_file] -query="#combine(ESSLLI Hamburg)"`

Exercises

- Compare results of different smoothing settings
  - extremes (e.g. Jelinek-Mercer: $\lambda = 0$ vs. $\lambda = 1$)
  - a sweep (e.g. Jelinek-Mercer: $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$)
- Report on (interesting) differences
- Where does changing parameter settings help? hurt? In terms of precision, recall or some average?
- Why?
- Be creative...