

Statistical Language Modeling for Information Access

Practical IV: Pseudo Relevance Feedback

Maarten de Rijke Edgar Meij Krisztian Balog

University of Amsterdam
Norwegian University of Science and Technology

August 1–4, 2011

Outline of the Course

- 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

Outline

1 Pseudo Relevance Feedback

- Lemur
- Indri

2 A Bit More On Evaluation

- Plotting
- Statistical Testing

3 Exercises

Lemur

- Lemur comes with five blind relevance feedback methods
- You can easily develop your own! Subclass `RetrievalMethod` and override the method `updateQuery` with your new method.
- All methods use, besides their own parameter values, two “global” parameter values:
 - ▶ `feedbackDocCount` – The number of documents to use for blind relevance feedback
 - ▶ `feedbackTermCount` – The number of terms to use

- To use them, put the `queryUpdateMethod` parameter in your retrieval parameter file
 - ▶ **mix** – Mixture
 - ▶ **div** – Divergence Minimization
 - ▶ **mc** – Markov Chain Translation Model
 - ▶ **rm1** – Relevance Model 1 (independent)
 - ▶ **rm2** – Relevance Model 2 (conditional)
- The first three use the parameter `feedbackMixtureNoise`
- Relevance models
 - ▶ only require the `feedbackDocCount` and `feedbackTermCount` parameters
 - ▶ require query-likelihood scores, so use
`<adjustedScoreMethod>q1</adjustedScoreMethod>`

Mixture

- Assume that each term w in the set of feedback documents \mathcal{F} is generated *independently* from some model R (Zhai and Lafferty, *CIKM* 2001):

$$P(\mathcal{F}|R) = \prod_i \prod_w P(w|R)^{n(w,d_i)}$$

- Mix in a background model

$$P(\mathcal{F}|R) = \prod_i \prod_w ((1 - \lambda)P(w|R) + \lambda P(w|GE))^{n(w,d_i)}$$

- Take the log to obtain the log-likelihood of the feedback documents:

$$\log P(\mathcal{F}|R) = \sum_i \sum_w n(w, d_i) \log((1 - \lambda)P(w|R) + \lambda P(w|GE))$$

Mixture

- Use an EM algorithm to determine $P(w|R)$

$$t(w) = n(w_i, d) \cdot \frac{(1 - \lambda)P(w|R)}{(1 - \lambda)P(w|R) + \lambda P(w|GE)}$$

$$P(w|R) = \frac{\sum_d t(w)}{\sum_{w'} \sum_d t(w')}$$

- Difference with yesterday's EM formula?
- `feedbackMixtureNoise = λ`
- One more parameter for this method: `emIterations`, which is the maximum number of iterations the EM algorithm will run (default: 50)
- The EM algorithm terminates earlier if the log-likelihood converges

Divergence Minimization

- Minimize the divergence between the query model and each feedback document $d \in \mathcal{F}$ (Zhai and Lafferty, *CIKM* 2001):

$$D_c(M; \mathcal{F}, GE) = \frac{1}{|\mathcal{F}|} \sum_{i=1}^n D(M || \hat{M}_{d_i}) - \lambda D(M || P(\cdot | GE))$$

- This query model will give the *best average score over the feedback documents*
- After some rewriting, we obtain

$$P(w|R) \propto \exp \left(\frac{1}{1-\lambda} \frac{1}{|\mathcal{F}|} \sum_i \log P(w|M_{d_i}) - \frac{1}{1-\lambda} \log P(w|GE) \right)$$

- `feedbackMixtureNoise = λ`

Markov Chain Translation Model

- Lafferty and Zhai, *SIGIR* 2001
- Inspired by Berger and Lafferty, *SIGIR* 1999
- Imagine a user looking to formulate a query for an information need
- She “surfs” the index in the following random manner
 - ▶ A word w_0 is chosen
 - ▶ A document d_0 containing that word is chosen (a choice which will be influenced by the number of times the word appears in d_0)
 - ▶ From that document, a new word w_1 is sampled
 - ▶ A new document containing w_1 is chosen
 - ▶ Etc.
 - ▶ After each step, there is some chance the user will stop browsing with probability α

Markov Chain Translation Model

- The posterior probability of sampling document d_i after term w_i is

$$P(d_i|w_i) = \frac{P(w_i|d_i)P(d_i)}{\sum_d P(w_i|d)P(d)}$$

- Having chosen a document d_i , a new word w_{i+1} is sampled from it according to $P(\cdot|d_i)$
- If we have evidence in the form of (pseudo-)relevant documents, we can use this approach to sample expansion terms:

$$P(w|q, \mathcal{F}) \propto P(w) \sum_{d \in \mathcal{F}} P(d|w)P(q|d)$$

- `feedbackMixtureNoise` = $1 - \alpha$ (!)

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>
<feedbackDocCount>10</feedbackDocCount>
<feedbackTermCount>5</feedbackTermCount>
<queryUpdateMethod>rm1</queryUpdateMethod>
<adjustedScoreMethod>q1</adjustedScoreMethod>
</parameters>
```

Interpolation

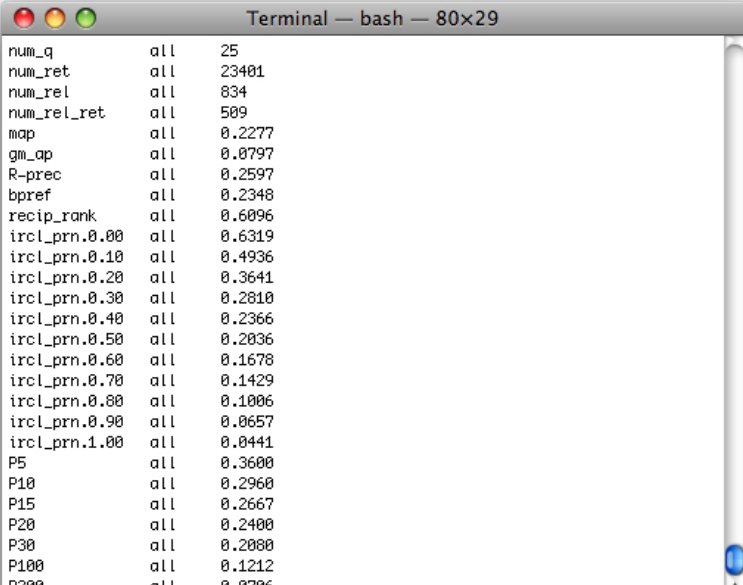
- After generating the expanded model using one of the methods, the original query may be blended in
- The following parameter values apply:
 - ▶ `feedbackCoefficient` – the coefficient of the feedback model for interpolation
 - ★ The value ranges from 0-1
 - ★ 0 meaning using only the original model (thus no updating/feedback)
 - ★ 1 meaning using only the feedback model (thus ignoring the original model)
 - ▶ `feedbackTermCount` – Truncate the feedback model to no more than a given number of terms
 - ▶ `feedbackProbThresh` – Truncate the feedback model to include only words with a probability higher than this threshold (default 0.001)
 - ▶ `feedbackProbSumThresh` – Truncate the feedback model until the sum of the probability of the included words reaches this threshold (default 1)

- Indri only implements relevance model 1 (independent)
- The initial retrieval of pseudo-relevant documents is **always** done using Dirichlet smoothing
- Specify one of the parameter settings below to enable feedback
 - ▶ `fbDocs` – number of feedback documents
 - ▶ `fbTerms` – number of terms to use
 - ▶ `fbOrigWeight` – interpolation weight of the original query (note that this is the inverse of Lemur!)

Indri Example

```
<parameters>
<index>/path/to/your/index</index>
<rule>method:jm,lambda:0.15</rule>
<count>1000</count>
<trecFormat>1</trecFormat>
<fbTerms>5</fbTerms>
<fbDocs>10</fbDocs>
<fbOrigWeight>0.2</fbOrigWeight>
</parameters>
```

trec_eval

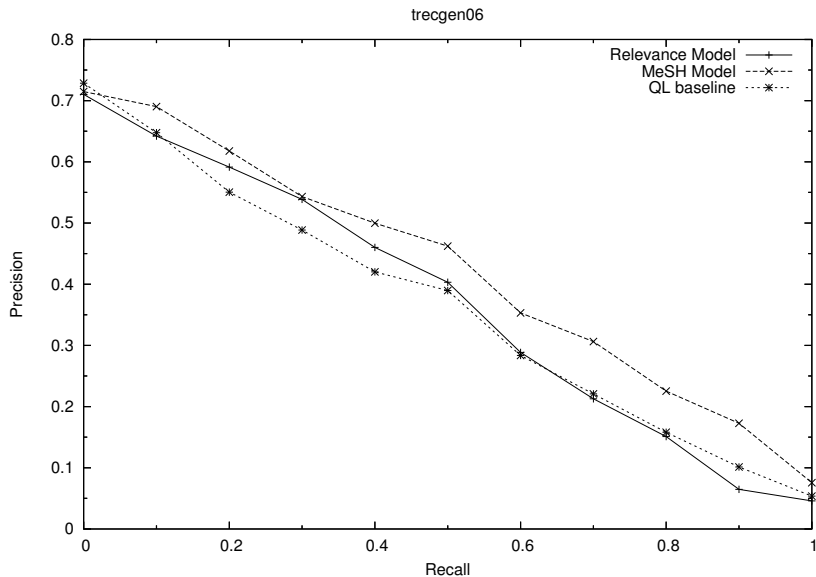


num_q	all	25
num_ret	all	23401
num_rel	all	834
num_rel_ret	all	509
map	all	0.2277
gm_ap	all	0.0797
R-prec	all	0.2597
bpref	all	0.2348
recip_rank	all	0.6096
ircl_prn.0.00	all	0.6319
ircl_prn.0.10	all	0.4936
ircl_prn.0.20	all	0.3641
ircl_prn.0.30	all	0.2810
ircl_prn.0.40	all	0.2366
ircl_prn.0.50	all	0.2036
ircl_prn.0.60	all	0.1678
ircl_prn.0.70	all	0.1429
ircl_prn.0.80	all	0.1006
ircl_prn.0.90	all	0.0657
ircl_prn.1.00	all	0.0441
P5	all	0.3600
P10	all	0.2960
P15	all	0.2667
P20	all	0.2400
P30	all	0.2080
P100	all	0.1212
P200	all	0.0706

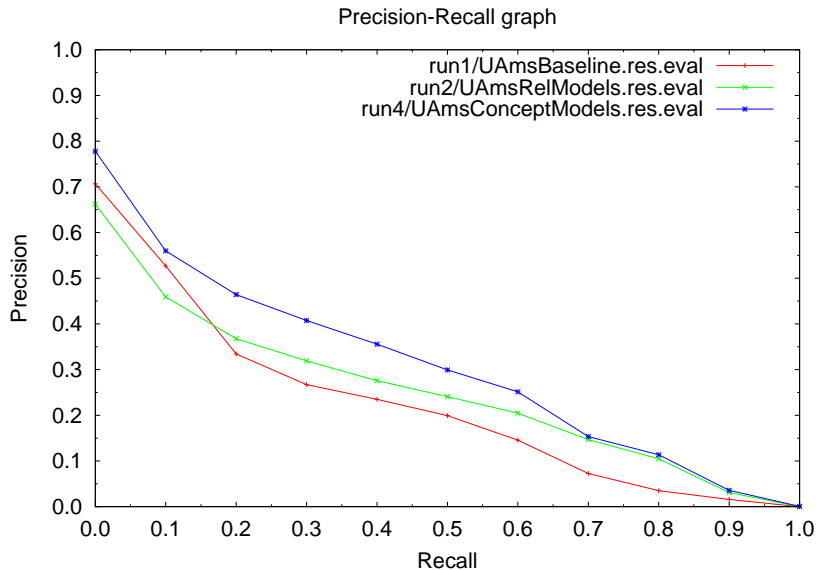
Plotting

- Scripts on the course web site
 - ▶ Requires Perl and Gnuplot
 - ▶ If you run Windows, consider using cygwin
- Precision-Recall graphs
 - ▶ Displays precision at fixed recall points
 - ▶ Used often in IR evaluations
 - ▶ Script: `ProcessPR.pl [trec_eval_output_files]`

Precision-Recall graphs



Precision-Recall graphs



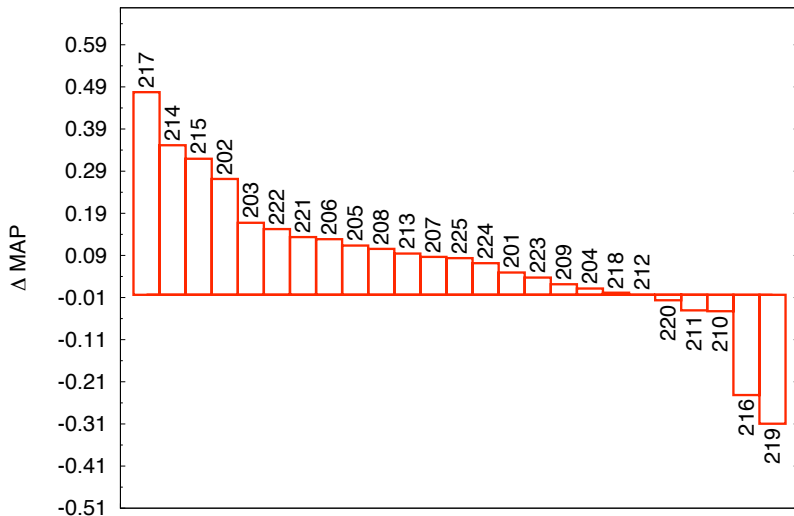
Plotting

- Per-topic differences

- ▶ Displays a per-topic comparison between two runs on a chosen measure
- ▶ Useful for determining which topics are hurt/helped as well as to get a general feel for how many topics were helped/hurt
- ▶ Script: `plotRunDifference.pl <measure>`
`<trec_eval_output_file_1> <trec_eval_output_file_2>`

Per-topic differences

run1/UAMsBaseline.res.eval - run4/UAMsConceptModels.res.eval



Statistical Testing

- Which test to use?
 - ▶ Sign test (weakest)
 - ▶ Wilcoxon
 - ▶ Paired t-test (strongest)
- Paired t-test script for comparing two retrieval runs on the course web site

Outline

1 Pseudo Relevance Feedback

- Lemur
- Indri

2 A Bit More On Evaluation

- Plotting
- Statistical Testing

3 Exercises

Exercises

- Choose the KL-divergence retrieval model, a smoothing method, and a set of smoothing parameters
- Compare results of different relevance feedback approaches
 - ▶ Different strategies
 - ▶ Different settings
- Report on (interesting) differences
- Plot the results
- Perform statistical tests to determine whether two runs are significantly different
- Where does relevance feedback help? hurt? In terms of precision, recall or some average? Why?