Language models applied to the field of Information Retrieval

Alejandro Bellogin Kouki
alejandro.bellogin@uam.es

February 19, 2008

Outline

Language Model: definition

- In speech recognition: probability distribution that captures statistical regularities of the generation of language
- Language models (for speech) attempt to predict the probability of the next word in an ordered sequence
- In IR: generation of queries $\rightsquigarrow$ random process $\Rightarrow$ model occurrences at document level without regard to sequential effects

Language Model: definition

- In speech recognition: probability distribution that captures statistical regularities of the generation of language
- Language models (for speech) attempt to predict the probability of the next word in an ordered sequence
- In IR: generation of queries $\rightsquigarrow$ random process $\Rightarrow$ model occurrences at document level without regard to sequential effects
- **Unigram** language models $\Longrightarrow$ terms are independent from each other

Language Model: definition

- In speech recognition: probability distribution that captures statistical regularities of the generation of language
- Language models (for speech) attempt to predict the probability of the next word in an ordered sequence
- In IR: generation of queries $\rightsquigarrow$ random process $\Rightarrow$ model occurrences at document level without regard to sequential effects
- **Unigram** language models $\Longrightarrow$ terms are independent from each other
- Phrases? Not unigram (but it seems possible)

Outline

Applications

- Ranking

Applications

- Ranking
- Prediction of query performance: It has received several names in different contexts and with distinct nuances:
  - Query ambiguity, vagueness
  - Query clarity (or lack of ambiguity), coherence
  - Query difficulty, performance, hardness

Technology and tools I

Language models framework is available in the following tools:

- Lemur
- Terrier

Some useful characteristics of each tool:

- Lemur includes a query clarity score calculator, so that it is very easy to get a score for some query and an indexed collection. One disadvantage is that we have not been able to create indexes programmatically and we need to use a graphical interface provided for this purpose. Another one is that the score returned by the tool is not the standard one explained here.

Technology and tools II

- Terrier is very easy to personalise, you can create your own collection decoder (for indexing) and your own weighting model. One possible application for this is to create a program that calculates clarity score using Terrier.
- Terrier supports several weighting models and it is easy to make them work. Besides that, it uses a very complete configuration file, which makes possible to have a personalised application without even write a line.
- Terrier handles all currently available TREC test collections what makes easier experimentation and comparation.

Technology and tools III

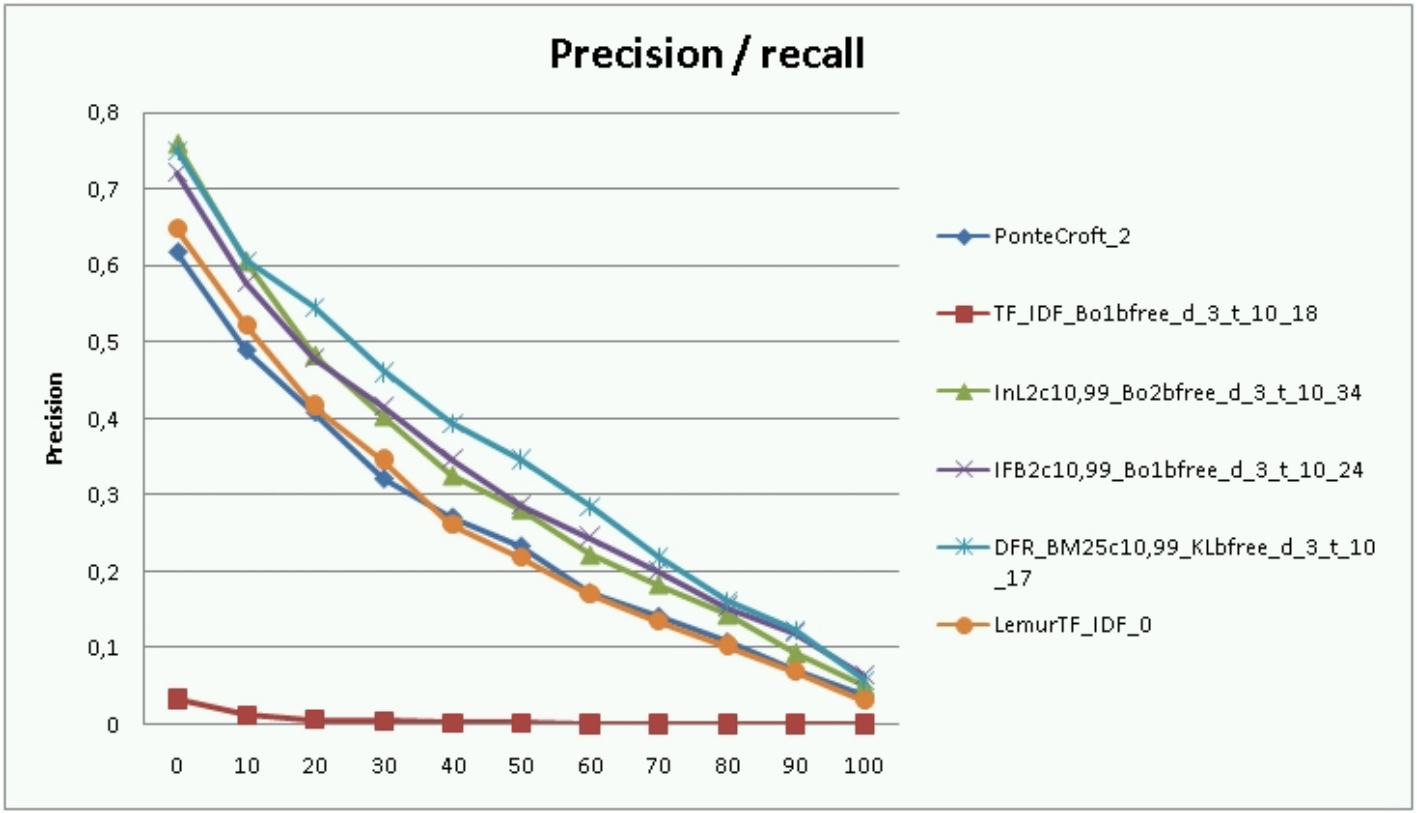

Figure: Some weighting models tested with the WT2G TREC collection

Ranking with language models I

1. To infer a language model for each document and to estimate the probability of generating the query according to each of these models
2. The documents are ranked according to these probabilities

This is known as the *query likelihood ranking principle*.

Ranking with language models II

Advantages:

- The data model and the discriminant function for retrieval are the same. In other works the document indexing and document retrieval are different, and several researchers had the goal of integrating both models.
- Collection statistics (term frequency, document length, document frequency) are integral parts of the language model and are not used heuristically.

Query performance

A query returning a mix of articles about different topics (it has low coherence) has a model more like the model of the collection as a whole, and it would get a low *clarity score* [5].

Examples I

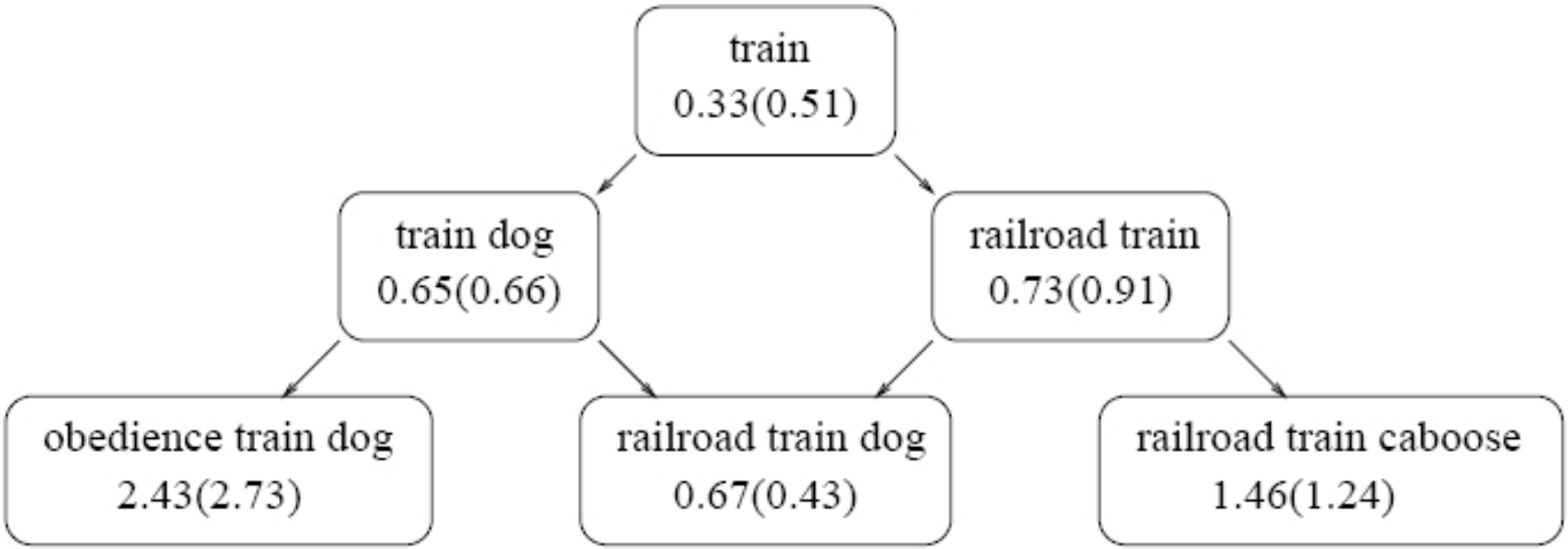

Figure: Clarity scores for some related queries using *Model 1* and *Model 2* (in parenthesis)

Examples II

- We have a document with two words: $D = \{\text{one}, \text{two}\}$. Given the next queries:

$$Q_1 = \{\text{one}\}, Q_2 = \{\text{two}\}, Q_3 = \{\text{three}\},$$

$$Q_4 = \{\text{three}, \text{two}\}, Q_5 = \{\text{one}, \text{two}\}$$

Examples II

- We have a document with two words: $D = \{\text{one}, \text{two}\}$. Given the next queries:

$$Q_1 = \{\text{one}\}, Q_2 = \{\text{two}\}, Q_3 = \{\text{three}\},$$

$$Q_4 = \{\text{three}, \text{two}\}, Q_5 = \{\text{one}, \text{two}\}$$

Lemur gives the next clarity scores:

$$cs(Q_1) = 1, cs(Q_2) = 1, cs(Q_3) = 0, cs(Q_4) = 1, cs(Q_5) = 1$$

Examples II

- We have a document with two words: $D = \{\text{one}, \text{two}\}$. Given the next queries:

$$Q_1 = \{\text{one}\}, Q_2 = \{\text{two}\}, Q_3 = \{\text{three}\},$$

$$Q_4 = \{\text{three}, \text{two}\}, Q_5 = \{\text{one}, \text{two}\}$$

Lemur gives the next clarity scores:

$$cs(Q_1) = 1, cs(Q_2) = 1, cs(Q_3) = 0, cs(Q_4) = 1, cs(Q_5) = 1$$

We obtain the following results by hand:

$$cs_1(Q_1) = -0.5, cs_1(Q_2) = -0.5, cs_1(Q_3) = 0,$$

$$cs_1(Q_4) = -0.5, cs_1(Q_5) = -0.5,$$

$$cs_2(Q_1) = 0, cs_2(Q_2) = 0, cs_2(Q_3) = 0,$$

$$cs_2(Q_4) = 0, cs_2(Q_5) = 0$$

Examples III

- Expansion:

$$
\begin{aligned}
P_{ml}(w|D) &= \frac{c(w;D)}{\sum_w c(w;D)} \\
P_{coll}(w) &= \frac{\sum_D c(w;D)}{\sum_D \sum_w c(w;D)} \\
P(w|D) &= \lambda P_{ml}(w|D) + (1-\lambda) P_{coll}(w) \\
cs_i(Q) &= \sum_{w \in V} P_i(w|Q) \log_2 \frac{P_i(w|Q)}{P_{coll}(w)}, i = 1, 2
\end{aligned}
$$

Examples IV

- If we use Bayesian inversion we will call it as cs_2, and cs_1 if we do not use it:

$$
\begin{aligned}
P(Q|D) &= \prod_{q \in Q} P(q|D) \\
P_1(w|Q) &= \sum_{D \in R} P(w|D)P(Q|D) \\
P_2(w|Q) &= \sum_{D \in R} P(w|D)P(D|Q)
\end{aligned}
$$

$$
P(D|Q) = \frac{P(D)}{P(Q)} P(Q|D)
$$

$$
P(Q) = \sum_{D \in \mathcal{C}} P(D)P(Q|D)
$$

Examples V

- In this special case:

$$
\begin{aligned}
|\mathcal{C}| &= 1 \Rightarrow P_{ml}(w|D) = P_{coll}(w) \\
|\mathcal{C}| &= 1 \Rightarrow P(D|Q) = \frac{P(D)}{P(D)P(Q|D)} P(Q|D) = 1 \Rightarrow \\
&\Rightarrow P(w|Q) = P(w|D) = P_{coll}(w) \Rightarrow \log_2 \frac{P(w|Q)}{P_{coll}(w)} = 0
\end{aligned}
$$

Why are not they the same? Because Lemur...

- Calculates the score over the query terms instead of over all terms in the vocabulary, i.e.

$$cs(Q) = \sum_{w \in V} P(w|Q) \log_2 \frac{P(w|Q)}{P_{coll}(w)}$$

$$cs_{\text{Lemur}}(Q) = \sum_{w \in Q} P(w|Q) \log_2 \frac{P(w|Q)}{P_{coll}(w)}$$

Why are not they the same? Because Lemur...

- Calculates the score over the query terms instead of over all terms in the vocabulary, i.e.

$$cs(Q) = \sum_{w \in V} P(w|Q) \log_2 \frac{P(w|Q)}{P_{coll}(w)}$$

$$cs_{\text{Lemur}}(Q) = \sum_{w \in Q} P(w|Q) \log_2 \frac{P(w|Q)}{P_{coll}(w)}$$

- Gives a probability of 1 to terms involved in the calculation (these terms are in fact part of the query), so:

$$cs_{\text{Lemur}}(Q) = \sum_{w \in Q} \log_2 \frac{1}{P_{coll}(w)}$$

Outline

Ranking models

- Summary

Author	$P(w\|M_D)$	$P(Q\|M_D)$	**Smooth**
Croft	$P_{ml}(t,D) = \frac{tf_{(t,D)}}{dl_D}$	$\prod_{w \in Q} P(w\|M_D) \cdot \prod_{w \notin Q} (1 - P(w\|M_D))$	Linear
Hiemstra	$\alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i,d)}{\sum_t tf(t,d)}$	$\prod_w P(w\|M_D)^{q_w}$	Linear
Croft (Feedback)	$\lambda P_{ml}(w\|M_D) + (1-\lambda)P(w)$		Linear

- The rest of models differ in the smoothing applied

First model

For each document we rank according to

$$\prod_{t \in Q} P_{ml}(t, D) = \prod_{t \in Q} \frac{tf_{(t,D)}}{dl_D}$$

where $tf_{(t,D)}$ is the frequency of term t in document D and dl_D is the lenght of document D,

Hiemstra's model

Hiemstra [8] presented a model in which documents and queries are defined by an ordered sequence of single terms. It uses linear combination to avoid sparsity problem and the next formulae:

$$
\begin{aligned}
P(D|N) &= \sum_{\mathcal{T}} P(\mathcal{T}|N)P(D|\mathcal{T}), \mathcal{T} = (T_1, \cdots, T_l) \\
P(D|\mathcal{T}) &= C \cdot P(D) \prod_{i=1}^{l} P(T_i|D), \frac{1}{C} = \sum_{d} P(D = d|\mathcal{T}) \\
P(\mathcal{T}|N) &= \frac{1}{N_q} \\
P(D = d) &= \frac{1}{N_d} \\
P(T_i = t_i|D = d) &= \alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i, d)}{\sum_t tf(t, d)}
\end{aligned}
$$

Relevance models I

Lavrenko and Croft presented in [13] two methods to estimate the probabilities of words in the relevant class (*relevance models*) using the query alone and addressing synonymy and polysemy:

- Method 1: i.i.d. sampling. Query words q_i and words w in relevant documents are sampled identically and independently from a unigram distribution M_R. $\mathcal{M}$ represents some finite universe of unigram distributions, we pick a distribution $M \in \mathcal{M}$ with a probability $P(M)$, and sample from it $k+1$ times. Then the total probability of observing w together with $q_1, \cdots, q_k$ is:

$$\begin{aligned} P(w, q_1, \cdots, q_k) &= \sum_{M \in \mathcal{M}} P(M) P(w, q_1, \cdots, q_k | M) \\ &= \sum_{M \in \mathcal{M}} P(M) P(w|M) \prod_{i=1}^{k} P(q_i|M) \end{aligned}$$

Relevance models II

- Method 2: conditional sampling. We fix a value of w according to some prior $P(w)$, then we pick a distribution $M_i \in \mathcal{M}$ according to $P(M_i|w)$ and the sample query word q_i from M_i with probability $P(q_i|M_i)$ k times. We assume the query words are independent of each other.

$$\begin{aligned} P(w, q_1, \cdots, q_k) &= P(w) \prod_{i=1}^{k} P(q_i|w) \\ &= P(w) \prod_{i=1}^{k} \sum_{M_i \in \mathcal{M}} P(M_i|w) P(q_i|M_i) \end{aligned}$$

Here we are free to pick a separate M_i for every q_i. This method is less sensitive to the choice of our universe of distributions $\mathcal{M}$.

Relevance feedback and personalisation

- A modification of the model was proposed by Croft et al. [4].
- They view the query as a sample of text from a model of the information need $\Rightarrow$ allows the queries to be generated by information need language models associated with individual users $\Rightarrow$ personalisation and relevance feedback
- Users and documents can be represented as a mixture of topic language models generated from previous interactions.
- Once you have the query model, retrieval could then be done:
  - Ranking the documents according to their *probability of being generated* by the query model
  - The query model and the document models could be *directly compared* and documents ranked by the similarity of these models.

The problem in this case is that we have to estimate the query model from very limited data.

Risk minimization I

- Lafferty and Zhai in [12] used risk minimization, as well as a language model for each document and for each query.
- The query language model can be exploited to model user preferences, the context of a query, synonymy and word senses.
- They see the query (based on Bayesian decision theory) as an output of some probabilistic process associated with the user $\mathcal{U}$, and similarly, a document as the output of some probabilistic process associated with an author or document source $\mathcal{S}$.
- They use Markov chains on the inverted indices of a document collection. They also use the Markov chains method to expand the query.

Risk minimization II

- Scenario: The user wants to formulate a query for an information need (The user has an index available to be searched). The user surfs in the following random way:
  1. A word w_0 is chosen
  2. The index is consulted and a document D_0 containing that word is chosen (this selection may be affected by the number of times the word appears in D_0 or by extrinsic data).
  3. From that document a new word w_1 is sampled, and so on. In this manner, the probability of selecting a document D_i from the inverted list for w_i is

$$P(D_i|w_i) = \frac{P(w_i|D_i)P(D_i)}{\sum_D p(w_i|D)P(D)}$$

  given the document model $P(\cdot|D)$ and a prior distribution in documents $P(D)$.

Risk minimization III

- A possible action to the system corresponds to a list of documents to return to the user who has issued query q.
- Such an action has an associated **loss function**, and an expected *risk* of action.
- The Bayesian decision rule is then to present the document list having the least expected risk. In this general case, we can have the loss function depending only on relevance (relevance based) or on query and document models (distance based).
- The classical probabilistic retrieval model and the language modeling approach are special cases of the relevance based loss functions and the vector space model as an special case of the distance based.

Smoothing

- Solve the following problems:
  - We do not wish to assign a probability of zero to a document that is missing one or more of the query terms.
  - The fact that we have not seen a term does not make it impossible to appear in the language model associated with a document, i.e. we do not want $P(t|M_D) = 0$.
  - We only have a document sized sample from the distribution of M_D, instead of getting an arbitrary sized sample of data from it.

Why smoothing?

- It is necessary to smooth the estimators used for language modeling, since they will generally under-estimate the probability of any word unseen in the document.
- This is done removing a little bit of probability mass from the more common terms and collecting it on the unseen ones.
- The terms with low document frequency have to be smoothed, because their average probability is based on a small amount of data and could be sensitive to outliers.

Smoothing generalization

We have the following general form of a smoothed model [23]:

$$P(w|D) = \begin{cases} P_s(w|D) & \text{if word } w \text{ is seen} \\ \alpha_D P(w|\mathcal{C}) & \text{otherwise} \end{cases}$$

where $P_s(w|D)$ is the smoothed probability of a word seen in the document, $P(w|\mathcal{C})$ is the collection language model and α_D is a coefficient controlling the probability mass asigned to unseen words, so that all probabilities must sum to one. If $P_s(w|D)$ is given, we must have

$$\alpha_D = \frac{1 - \sum_{w:c(w;D)>0} P_s(w|D)}{1 - \sum_{w:c(w;D)>0} P_s(w|C)}$$

where $c(w; D)$ denotes the count of word w in D. Because of this, smoothing methods differ in their choice of $P_s(w|D)$.

Smoothing models

Method	$P_s(w\|D)$	α_D
Jelinek-Mercer (linear interpolation)	$(1-\lambda)P_{ml}(w\|D)+\lambda p(w\|\mathcal{C})$	λ
Dirichlet	$\frac{c(w;D)+\mu P(w\|\mathcal{C})}{\sum_w c(w;D)+\mu}$	$\frac{\mu}{\sum_w c(w;D)+\mu}$
Absolute discount	$\frac{\max(c(w;D)-\delta,0)}{\sum_w c(w;D)}+\frac{\delta\|D\|_u}{\|D\|}P(w\|\mathcal{C})$	$\frac{\delta\|D\|_u}{\|D\|}$
Good-Turing	$(1-\omega)P_{GT}(w\|D)+\omega P(w\|\mathcal{C})$	ω

Jelinek-Mercer smoothing

$$(1-\lambda)P_{ml}(w|D)+\lambda p(w|\mathcal{C})$$

- It uses a parameter to control the influence of each involved model (maximum likelihood and collection ones). It is based on the maximum likelihood estimator:

$$P_{ml}(w|D)=\frac{c(w;D)}{\sum_w c(w;D)}$$

Dirichlet smoothing

$$\frac{c(w;D) + \mu P(w|\mathcal{C})}{\sum_w c(w;D) + \mu}$$

- Dirichlet method stands for *Bayesian smoothing using Dirichlet priors*, where the language model is a multinomial distribution and the conjugate prior for Bayesian analysis is the Dirichlet distribution with parameters $(\mu P(w_1|\mathcal{C}), \cdots, \mu P(w_n|\mathcal{C}))$.
- The Laplace method is a special case of this technique (this one adds 1 to all frequency counts, it is very simple but it gives too much probability mass to unseen terms).

Absolute discount smoothing

$$\frac{\max\left(c(w;D)-\delta,0\right)}{\sum_w c(w;D)}+\frac{\delta|D|_u}{|D|}P(w|\mathcal{C})$$

- *Absolute discounting* method is similar to the first one, but it substracts a constant instead of multiplies by $1-\lambda$ for discounting the seen word probability. In this model $|D|_u$ is the number of unique terms in document D and $|D|=\sum_w c(w;D)$, i.e. the total count of words in the document.

Good-Turing estimator I

- *Good-Turing* [18] needs the expected value of the number of terms with some specified frequency in a document, which is almost impossible or very expensive to compute. The original formula is

$$tf^* = (tf + 1)\frac{E(N_{tf+1})}{E(N_{tf})}$$

- Instead of trying to calculate it, Song and Croft used a smoothed function (curve-fitting function) for the expected value:

$$P_{GT}(w|D) = (tf + 1)\frac{S(N_{tf+1})}{S(N_{tf})N_D}$$

Good-Turing estimator II

- They used the interpolation method with this estimator instead of the maximum likelihood model:

$$(1-\omega)P_{GT}(w|D)+\omega P(w|\mathcal{C})$$

and also a *weighted product* approach:

$$P(w|D)=P_{GT}(w|D)^{\omega}+P(w|\mathcal{C})^{(1-\omega)}$$

- One advantage of the interpolation method is that the probabilities are normalized so that the total probability mass is 1, whereas in the weighted product method it does not happen.

Katz smoothing I

- *Katz smoothing* [11] extends the previous estimator. At first it was used for speech recognition and it treated differently words of different count by means of the next formula (for trigram model for convenience [6]):

$$P_{Katz}(w_i|w_{i-2}^{i-1}) = \begin{cases} C(w_{i-2}^{i})/C(w_{i-2}^{i-1}) & \text{if} \quad r > r_t \\ d_{r,3}C(w_{i-2}^{i})/C(w_{i-2}^{i-1}) & \text{if} \quad 0 < r \leq r_t \\ \alpha(w_{i-2}^{i-1})P_{Katz}(w_i|w_{i-1}) & \text{if} \quad r = 0 \end{cases}$$

Katz smoothing II

- Where

$$
\begin{aligned}
C(\cdot) &= \text{count of the event} \\
r &= C(w_{i-2}^{i}) \\
r_t &= \text{threshold for discounting purpose} \\
\alpha(w_{i-2}^{i-1}) &= \frac{1 - \sum_{w_i : r>0} P_{Katz}(w_i | w_{i-2}^{i-1})}{1 - \sum_{w_i : r>0} P_{Katz}(w_i | w_{i-1})} \\
d_{r,3} &= \frac{\frac{r^*}{r} - \frac{(r_t+1)n_{r_t+1}}{n_1}}{1 - \frac{(r_t+1)n_{r_t+1}}{n_1}}
\end{aligned}
$$

where n_r is the number of n-grams ocurring exactly r times. We have to note that the key factor in Katz smoothing is $d_{r,n}$, because every n-gram units occuring exactly r times will be treated as $d_{r,n} \cdot r$ times, or equivalently, it will be removed $(1 - d_{r,n}) \cdot r$ times from every such units and redistributed to the unseen ones.

Semantic smoothing

- Berger and Lafferty [3] tried to incorporate some kind of semantic smoothing into the language modeling approach by means of estimating translation models $t(q|w)$ for mapping a document term w to a query term q. The document-to-query model becomes

$$P(q|D) = \prod_{i=1}^{m} \sum_{w} t(q_i|w) P(w|D)$$

- In this smoothing method synonyms and word sense information is incorporated into the models [12]. The goal is that a document containing the term $w =$ automobile may be retrieved to answer a query that includes a term $q =$ car, even if the query term does not appear in the document.

Smoothing conclusions

- According to [23] there is a strong (and unexpected) correlation between the effect of smoothing and the type of query. This leads the authors to guess that smoothing plays two roles:
  - One role is to improve the accuracy of the estimated document language model (*estimation role*)
  - The other role is to accommodate generation of common and non-informative words in a query (*query modeling*).
- In their experiments they found that title queries were more dominated by the estimation role, since these queries have few or no non-informative common words, whereas for long queries, the effect was more influenced by the role of query modeling.
- Linear interpolation method for smoothing gives a robust estimation of common, context-free words, treated as *stopwords* in IR systems.

Outline

Clarity score I

- First formulation:

$$
\begin{aligned}
P(w|Q) &= \sum_{D \in R} P(w|D)P(D|Q), \quad P(Q|D) = \prod_{q \in Q} P(q|D) \\
P(w|D) &= \lambda P_{ml}(w|D) + (1-\lambda)P_{coll}(w) \\
\text{clarity score} &= \sum_{w \in V} P(w|Q) \log_2 \frac{P(w|Q)}{P_{coll}(w)}
\end{aligned}
$$

with w any term, Q the query, D a document or the model, R is the set of documents that contain at least one query term, $P_{ml}(w|D)$ is the relative frequency of term w in documents D, $P_{coll}(w)$ is the relative frequency of the term in the collection as a whole, λ is a parameter (in their work, 0.6) and V is the entire vocabulary. $P(D|Q)$ is the Bayesian inversion of $P(Q|D)$ with uniform document priors.

Clarity score II

- The score is simply calculated through the Kullback-Leibler divergence or relative entropy between the query and collection language models, which is a natural approach since entropy measures how strongly a distribution specifies certain values, in this case terms.
- In terms of coding theory, clarity is the average number of bits that would be wasted by encoding word events from the collection model with a code that was optimally designed for the query model [4].

Clarity score III

- The authors used the Lavrenko and Croft's *Method* 1, but they point out that using *Method* 2 seems to *punish* more harshly the addition of a term that does not co-occur with the other query terms in documents in the collection.
- This may be caused by the stronger independence assumptions employed in this method, because it will assign low probability estimations for terms not appearing in the query (although they could be in the same document as a query term) to the contrary what *Model* 1 does.
- We can see in figure 2 how this score works with some related queries. It is worth noting that the query created from the combination of the two presented meanings receives the lowest score only with *Method* 2.

Clarity score: correlation

- There is a strong correlation between the clarity score of a test query with respect to the appropriate test collection and the performance of that query.
- This may be due to the fact that a low-coherence retrieval is likely to contain many irrelevant documents in the top ranks and a high-coherence retrieval often contains many relevant documents in the top ranks.

Clarity score: variants I

- Turpin and Hersh [19] modified this definition and used instead of the one written before, the next

$$P(w|Q) = \sum_{D \in R} P(w|D)P(Q|D)$$

- Their experiment:
  - The second query issued by a user generally had a higher clarity score than the first, but after that next queries did not improve in clarity score.
  - Another observation is that experienced searchers (in their experiment, librarians) improve queries with repeated searches as much as less experienced searchers (postgraduate students).

Clarity score: variants II

- Croft et al. [4] replaced the distribution $P(D|Q)$ with an approximation that uses a fixed number N of documents rather than the entire retrieved set.
- This modification results in large performance gains, since the system will only have to mix N documents as maximum in the computation.

Clarity score: variants III a

- Ounis and He [7] studied some predictors for query performance.
- They also introduced a simplification of the clarity score given by

$$
\begin{aligned}
SCS &= \sum_{Q} P_{ml}(w|Q) \log_2 \frac{P_{ml}(w|Q)}{P_{coll}(w)} \\
P_{ml} &= \frac{qtf}{ql} \\
qtf &= \text{number of occurences of a query term in the query} \\
ql &= \text{query length}
\end{aligned}
$$

Clarity score: variants III b

- They simplified the score avoiding the computation of relevance scores for the query model, which is time-consuming.
- They also smoothed this estimator assuming an increasing query term frequency
- They analyse the linear and non-parametric (using Spearman's rank) correlation, and they found that among six predictors, the simplified definition of clarity score (SCS) and the average inverse collection term frequency have the **strongest correlation** with average precision (representing the query performance in all the experiments) for short queries. SCS is also one of the most correlated with average precision for normal and long queries. If we consider the smoothed version it was improved significantly for normal and long queries. Besides this, they found that the use of two different document weighting models, i.e. PL2 and BM25, does not affect the correlations of the proposed predictors with average precision

Clarity score: variants IV

- Other authors used clarity score to present a list of clarifying options to the searcher helping her in ambiguous queries [1]. They achieved this using *part-of-speech patterns*.
- Qiu et al [16] quantify query ambiguity based on the topic structure selected from the ODP taxonomy (Open Directory Project). They found a strong positive association between their measure for ambiguity quantification and precision. The measure used was

$$\text{clarity score} = F(|\{\text{topics}\}_{\text{intersect}}|)$$

where $F(x)$ is a function whose values decreases as its argument increases, such as $F(x) = \frac{1}{x+1}$, the one adopted in their experiment.

Clarity score: variants V

- Winaver et al. proposed in [21] an approach for query expansion in which, given a query, they create a set of language models representing different forms of its expansion by varying the parameters' values of some expansion method, then, they select a single model using some criteria.
- One of the criteria they suggest is to calculate the clarity score of each model and to **select the one maximising this score**. Once a query model is selected, they rank documents according Kullback-Leibler measure with respect to the collection model.

Outline

Future work

- We plan to develop the following ideas as soon as possible:
  - Given a collection with a list of predefined queries (such as TREC), calculate the query clarity for each query and split the queries according to their score. Then, we can calculate somehow (parametric vs non-parametric estimation) a distribution for each segment of clarity. When a new query is received, its clarity score is calculated and the corresponding distribution is assigned.
  - For rank fusion, we may want to give more weight to the system that produces a higher clarity score given a query. Is this correct?

Future work

- We plan to develop the following ideas as soon as possible:
  - Given a collection with a list of predefined queries (such as TREC), calculate the query clarity for each query and split the queries according to their score. Then, we can calculate somehow (parametric vs non-parametric estimation) a distribution for each segment of clarity. When a new query is received, its clarity score is calculated and the corresponding distribution is assigned.
  - For rank fusion, we may want to give more weight to the system that produces a higher clarity score given a query. Is this correct?
- For making all of this possible, it would be interesting to have an application that calculates precisely the query clarity. We want to develop such an application within the Terrier platform, and use it to perform our evaluations, along with the robust track [20] as the test corpus.

Future work

- We plan to develop the following ideas as soon as possible:
  - Given a collection with a list of predefined queries (such as TREC), calculate the query clarity for each query and split the queries according to their score. Then, we can calculate somehow (parametric vs non-parametric estimation) a distribution for each segment of clarity. When a new query is received, its clarity score is calculated and the corresponding distribution is assigned.
  - For rank fusion, we may want to give more weight to the system that produces a higher clarity score given a query. Is this correct?
- For making all of this possible, it would be interesting to have an application that calculates precisely the query clarity. We want to develop such an application within the Terrier platform, and use it to perform our evaluations, along with the robust track [20] as the test corpus.
- Related with this topic is the study of entropy (entropy models) and its application to model vagueness. In the future we plan to study these subjects and find a closer relation with the technique explained here.

Outline


James Allan and Hema Raghavan.
Using part-of-speech patterns to reduce query ambiguity.
In *25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314, 2002.


Giambattista Amati, Claudio Carpineto, and Giovanni Romano.
Query difficulty, robustness, and selective application of query expansion.
Advances in Information Retrieval, pages 127–137, 2004.


Adam Berger and John Lafferty.
Information retrieval as statistical translation.
In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–229, New York, NY, USA, 1999. ACM Press.


Bruce W. Croft, Stephen Cronen-Townsend, and Victor Lavrenko.
Relevance feedback and personalization: A language modeling perspective.

In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.


Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft.
Predicting query performance.
In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, New York, NY, USA, 2002. ACM.


W. U. Genqing, Fang Zheng, W. U. Wenhu, X. U. Mingxing, and J. I. N. Ling.
Improved katz smoothing for language modeling in speech recognition.
pages 925–928, September 2002.


Ben He and Iadh Ounis.
Query performance prediction.
Information Systems, 31(7):585–594, November 2006.


Djoerd Hiemstra.
A linguistically motivated probabilistic model of information retrieval.

In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 569–584, London, UK, 1998. Springer-Verlag.

Djoerd Hiemstra.
Using language models for information retrieval.
PhD thesis, 2000.

Thomas Hofmann.
Probabilistic Latent Semantic Indexing.
In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

S. Katz.
Estimation of probabilities from sparse data for the language model component of a speech recognizer.
Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on, 35(3):400–401, 1987.


John Lafferty and Chengxiang Zhai.
Document language models, query models, and risk minimization for information retrieval.
In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001. ACM.


Victor Lavrenko and W. Bruce Croft.
Relevance based language models.
In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127, New York, NY, USA, 2001. ACM.


Jay M. Ponte.
A language modeling approach to information retrieval.
PhD thesis, Amherst, MA, USA, 1998.


Jay M. Ponte and W. Bruce Croft.
A language modeling approach to information retrieval.

In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.


Guang Qiu, Kangmiao Liu, Jiajun Bu, Chun Chen, and Zhiming Kang.
Quantify query ambiguity using odp metadata.
In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 697–698, New York, NY, USA, 2007. ACM.


H. Schutze and J. Pedersen.
Information retrieval based on word senses.
In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1995.


Fei Song and W. Bruce Croft.
A general language model for information retrieval.
In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.


A. Turpin and W. Hersh.
Do clarity scores for queries correlate with user performance?
In *ADC '04: Proceedings of the 15th Australasian database conference*, pages 85–91, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.


Ellen M. Voorhees.
The trec 2005 robust track.
SIGIR Forum, 40(1):41–48, June 2006.


Mattan Winaver, Oren Kurland, and Carmel Domshlak.
Towards robust query expansion: model selection in the language modeling framework.
In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 729–730, New York, NY, USA, 2007. ACM Press.


C. Zhai.
Risk Minimization and Language Modeling in Text Retrieval.

PhD thesis, 2002.

Chengxiang Zhai and John Lafferty.
A study of smoothing methods for language models applied to ad hoc information retrieval.
In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA, 2001. ACM Press.