Language models applied to the field of Information Retrieval

Alejandro Bellogin Kouki

February 18, 2008

1 Introduction

Information retrieval deals with the representation, storage, organization, and access to information items. The emphasis is on the retrieval of information as opposed to the retrieval of data. Nowadays, its most visible applications are (web) search engines.

In UAM's Master Degree in Computer Science there is a course named *Recuperación y almacenamiento de información en la web* in which some classic retrieval models are studied (Vector Space Model, Probabilistic Model, $\cdots$), along with fusion ranking techniques and other related concepts. Some new techniques are out of the scope of this course, such as Language Modelling, an innovative probabilistic-based framework used to rank documents and for query analysis.

Since it is a powerful and not very spread Information Retrieval model it is very interesting to learn its functionality and capabilities. Besides that, we may have here a possible application in the thesis' area. In particular, envisioned application to query characterization (difficulty, performance, ambiguity), which can be used, for instance, to explore novel approaches to dynamic rank fusion.

The following are the goals of this work:

- Learn the basic principles of LM
- Get a wide perspective of the state of the art in the area of LM
- Explore advanced applications and research topics
- Get familiarity with the tools through examples and tests

2 Language Models: a description

Language models in the context of the information retrieval task were born in 1998, when Ponte and Croft presented their proposal [18] (these models are slightly different from the ones used in statistical techniques for speech recognition). In that moment, some researches started to use this new approach, there was even a project (Mongrel: Supporting Effective Access Through User- and Topic-Based Language Models[1], 2001-2003) where prominent authors in

[1] http://www.scils.rutgers.edu/etc/mongrel/index.htm

the area like Croft, Allan or Belkin did some research about the use of language models within relevance feedback, personalisation or query analysis. Several thesis worked in this subject too ([10], [27], [17]).

In 1999 Croft improved his previous work, this time with Song [23]. In this paper the authors presented better smoothing techniques to use within language models. Two years later Lavrenko and Croft proposed a technique to estimate a relevance model without training data [15], avoiding heuristic approaches by previous researchers. The next year Cronen-Townsend, Zhou and Croft introduced the concept of *clarity score*, studying the relation between this definition and the prediction of a query performance.

Next sections tackle the aforementioned topics, focusing in the seminal ones and presenting some novel ideas arisen from these. We are going to center this work on the **unigram** language models, which assume terms are independent from each other.

2.1 Ranking with language models

In its first description [18], the approach consisted in infering a language model for each document and to estimate the probability of generating the query according to each of these models. Then the documents are ranked according to these probabilities (this is known as the *query likelihood ranking principle*). This idea has the following 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. Here the authors relax some assumptions made by these models about the data:
  - The parametric assumption
  - Documents are members of pre-defined classes
- Collection statistics (term frequency, document length, document frequency) are integral parts of the language model and are not used heuristically.

This ranking was firstly produced by the following formula:

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

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

To solve these problems we have to smooth the estimator and get it from a larger amount of data:

$$
\begin{aligned}
P(Q|M_D) &= \prod_{t \in Q} P(t|M_D) \times \prod_{t \notin Q} 1.0 - P(t|M_D) \\
P(t|M_D) &= \begin{cases} P_{ml}(t,D)^{(1.0-\hat{R}_{t,D})} \times P_{\text{avg}}(t)^{\hat{R}_{t,D}} & \text{if} \quad tf_{(t,D)} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{cases} \\
P_{\text{avg}}(t) &= \frac{\sum_{D_{(t \in D)}} P_{ml}(t|M_D)}{df_t} \\
\hat{R}_{t,D} &= \left(\frac{1.0}{1.0+\bar{f}_t}\right) \times \left(\frac{\bar{f}_t}{1.0+\bar{f}_t}\right)^{tf_{t,D}} \\
& \bar{f}_t = P_{\text{avg}}(t) \times dl_D \\
P_{ml}(t|M_D) &= \frac{tf_{(t,D)}}{dl_D}
\end{aligned}
$$

This estimation could be improved if the terms with low document frequency are smoothed, because their average probability is based on a small amount of data and could be sensitive to outliers.

There has been other improvements related with the use of other smoothing techniques. Now we continue with other models, discarding the analysis of those approaches until the next section, where they will be described.

Some authors utilised the same idea but changing the way in which documents are modeled:

- As mixtures of topics [11]
- Translation probabilities were introduced to deal with synonymy and cross-lingual retrieval [3]

At the same time, Hiemstra [9] 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:

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

Lavrenko and Croft presented in [15] 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:

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

- 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 $\mathcal{M}$.

Regarding the relevance feedback process, a modification of the model was proposed by Croft, Cronen-Townsend and Lavrenko [4]. In this work they view the query as a sample of text from a model of the information need, this new approach allows the queries to be generated by information need language models associated with individual users, making available personalisation and relevance feedback, since users, like 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 either by ranking the documents according to their *probability of being generated* by the query model, or 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.

Another approach given by Lafferty and Zhai [14] uses risk minimization, as well as a language model for each document and for each query. Besides that, the query language model can be exploited to model user preferences, the context of a query, synonymy and word senses. They introduce a novel method for estimating query models that uses Markov chains on the inverted indices of a document collection. This random walk

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

has a natural interpretation in terms of document language models, and results in practical and effective translation models and query language models. This

is because 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 $\mathcal{S}$. The query model could encode detailed knowledge about a user's information need and the context in which she makes the query. 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). They mention that the classical probabilistic retrieval model and the language modeling approach are special cases of the former and the vector space model as an special case of the latter. Random walk written before comes from the next scenario: the user wants to formulate a query for an information need, and suppose that the user has an index available to be searched. The user surfs in the following random way: a word w_0 is chosen, then 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). 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 the formula presented before, given the document model $P(\cdot|D)$ and a prior distribution in documents $P(D)$. They also use the Markov chains method to expand the query.

2.1.1 Smoothing techniques

We have seen that a very important piece in the language modeling framework is the choice of the smoothing technique. In [29] we can find an analysis about the influence these techniques have in the retrieval results.

As we have said previously 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. Actually we have the following general form of a smoothed model [29]:

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

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 $P_s(w|D)$. We can see a summary in the next table, where a larger parameter value means more smoothing.

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

The most basic smoothing is the linear interpolation or *Jelinek-Mercer method*, which 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 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 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.

In [29] the authors only analyse these methods because they are popular and relatively efficient to implement, unlike Katz or Good-Turing estimators because of using the count of words with the same frequency, which is expensive to compute. In this analysis they found the next behaviour using some TREC collections (TREC7 ad hoc, TREC8 ad hoc, TREC8 web track):

- The length normalization term is a constant in the Jelinek-Mercer because α_D is the same for all documents. The score can be interpreted as a sum of weights over each matched term. For title queries, the retrieval performance tends to be optimized when λ is small (around 0.1), whereas for long queries, the optimal point is generally higher, and usually around 0.7. This suggests that long queries need more smoothing, and less emphasis is placed on the relative weighting of terms.

- Dirichlet priors. α_D is document-dependent: it is smaller for long documents (length normalization component that penalizes long documents); as μ gets large, α_D tends to 1, and all term weights tend to zero. However, the optimal value of μ does not seem to be much different for title queries and long queries but vary from collection to collection.

- Absolute discounting. α_D is also document sensitive. The optimal value of δ does not seem to be much different for title queries and long queries, but this is true across all testing collections.

Concerning the rest of the interpolation methods, *Good-Turing* [23] 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:

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

They used the interpolation method with this estimator instead of the maximum likelihood model (which is the one written in the table) 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 [12] 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 [7]):

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

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.

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 [14]. 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.

Now we can see some conclusions about these estimators. According to [29] 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.

We can also say that linear interpolation method for smoothing gives a robust estimation of common, context-free words, treated as *stopwords* in IR systems.

2.2 Query performance

Three years after the development of the language model theory already explained, some authors proposed to use language models for query analysis. Actually, this analysis was focused on quantifying the ambiguity of the query, but it has received several names in different contexts and with distinct nuances:

- Query ambiguity
- Query vagueness
- Query clarity (or lack of ambiguity)
- Query coherence
- Query difficulty
- Query performance
- Query hardness

It may sound strange to connect performance with ambiguity, but it makes sense. Actually there are several experiments demonstrating that query ambiguity is correlated with query performance. Because of that, everything comes down to obtain the quantity of ambiguity a query has in the better possible way. In the last years, there has been a lot of works about this problem, and in SIGIR 2005 even a workshop was celebrated under the name *Predicting Query Difficulty - Methods and Applications*[2]

There have been some authors that addressed this goal [2, 20], but Steve Cronen-Townsend, Yun Zhou and Bruce Croft were the first that managed this

[2] http://www.haifa.il.ibm.com/sigir05-qp/index.html

within the language model framework [6]. They proposed that 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 score in what they defined as *clarity score*. They used the following 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. 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. Interpreted 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].

In this formulation 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 1 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.

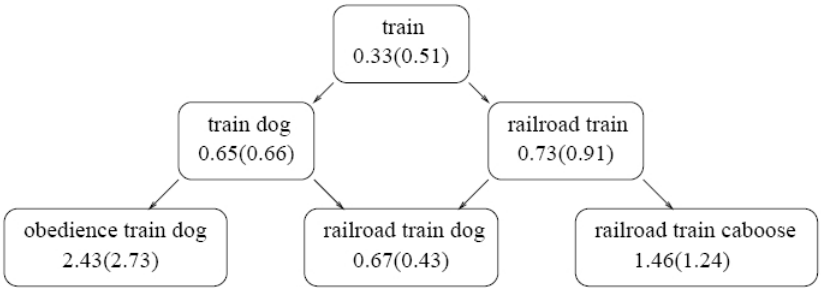

Figure 1: Clarity scores for some related queries using *Model 1* and *Model 2* (in parenthesis)

From their study one conclusion arises: 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.

Turpin and Hersh [24] modified this definition and used instead of the one written before, the next

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

They verify that there is no correlation between user performance and clarity score for their experiments. This is unsurprising, because we have already presented a strong correlation between MAP and clarity scores, and there is no correlation between MAP and user performance in data they used.

Something more interesting is their analysis of the change in clarity of user's queries over time. 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).

In other work, 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.

Ounis and He [8] studied some predictors for query performance. They also introduced a simplification of the clarity score given by

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

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[3].

In their work 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

[3]This is done in the following way: let $\rho = C \cdot ql^{\beta}$ the query length density function, with $\beta > 0$. They substitute the qtf factor by $qtfn$ calculated as

$$qtfn = qtf \cdot \int_{ql}^{ql+avg_{ql}} \rho d(ql) = qtf \cdot v \cdot ((ql + avg_{ql})^{1.5} - ql^{1.5})$$

where v is a free parameter and avg_{ql} is the average query length

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

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.

More recently, Qiu et al [19] used the Cronen-Townsend's ideas to 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.

Winaver et al. proposed in [26] 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.

3 Technology and tools

In the field of Information Retrieval there are several tools to index and/or retrieve documents, some of them use algebraic models (Lucene[4]), others use probabilistic ones (Lemur[5]) and others support a few different models (Terrier[6]).

Language models framework is available in the following tools:

- Lemur: written in C++ it supports XML and structured document retrieval, construction of simple language models for documents and queries and it has been used for 6 years by a large user community. It supports relevance feedback, smoothing (Dirichlet priors or Marchov Chains) and document priors (such as PageRank).

- Terrier: written in Java it indexes common file formats such as HTML, PDF or Microsoft Word. It provides many standard document weighting models, including DFR (Divergence From Randomness), Okapi BM25, language modelling and TF-IDF.

We have tested some applications in the tools mentioned before, and we have been able to found 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

[4]http://lucene.apache.org
[5]http://www.lemurproject.org
[6]http://ir.dcs.gla.ac.uk/terrier

is that the score returned by the tool is not the standard one explained here.

- 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.

In figure 2 we can see some results given by Terrier about the precision/recall curves with different weighting models (including the Lemur version of tf-idf, whose code is herewith presented)

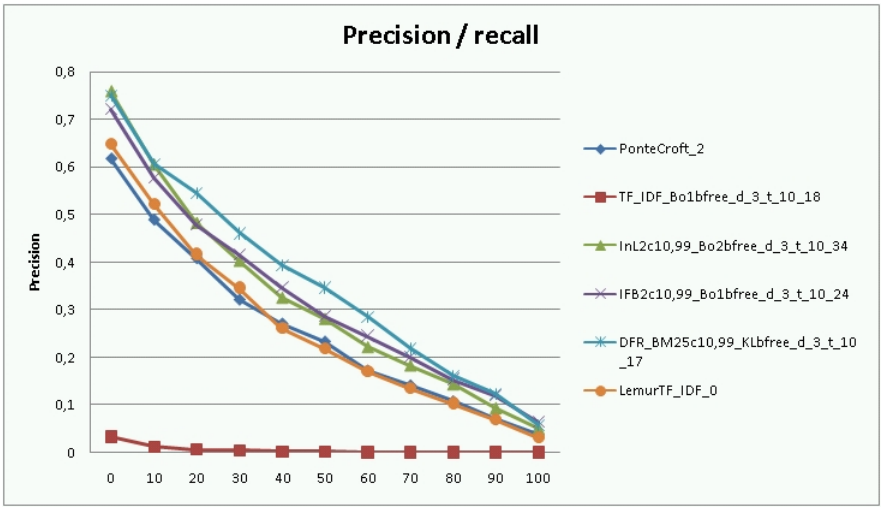

Figure 2: Some weighting models tested with the WT2G TREC collection

Lemur's tf-idf:

```
public final double score(double tf, double docLength) {
        double Robertson_tf = k_1*tf/(tf+k_1*(1-b+b*
            docLength/averageDocumentLength));
        double idf = i.log(numberOfDocuments/
            documentFrequency);
        return keyFrequency*Robertson_tf * Math.pow(idf,
            2);
}
```

Normal tf-idf:

```
public final double score(double tf, double docLength) {
```

```
        double Robertson_tf = k_1*tf/(tf+k_1*(1-b+b*
            docLength/averageDocumentLength));
        double idf = i.log(numberOfDocuments/
            documentFrequency+1);
        return keyFrequency * Robertson_tf * idf;
}
```

4 Examples

We have used Lemur tool to calculate easily clarity score in different situations, appart from that we have done the calculations by hand in some simple examples.

Using Lemur you can find these results doing the next steps:

1. Create the document collection
2. Index the collection (we have used the graphic interface `IndexUI.jar` included in the installation of Lemur)
3. Create two more files: a file containing query(ies) to be executed and a configuration file with the index path, query file and other parameters
4. Execute the query clarity application provided by Lemur

4.1 First example

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:

$$cs_1(Q_1) = -0.5, cs_1(Q_2) = -0.5, cs_1(Q_3) = 0, cs_1(Q_4) = -0.5, cs_1(Q_5) = -0.5,$$

$$cs_2(Q_1) = 0, cs_2(Q_2) = 0, cs_2(Q_3) = 0, cs_2(Q_4) = 0, cs_2(Q_5) = 0$$

after the next expansion:

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

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

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

If we examine formulae for the cs_2 it is easy to realise why in this example it is always 0:

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

Conclusion: if you only have one document and you use Bayesian inversion, clarity score does not work.

We can look now clarity score for the first query calculated without inversion ($\lambda = 0.6$):

$$\begin{aligned} V &= \{\text{one}, \text{two}\} \\ P_{coll}(\text{one}) &= P_{ml}(\text{one}|D) = \frac{1}{2} \Rightarrow P(\text{one}|D) = \frac{1}{2} \ \ \forall\lambda \\ P_{coll}(\text{two}) &= P_{ml}(\text{two}|D) = \frac{1}{2} \Rightarrow P(\text{two}|D) = \frac{1}{2} \ \ \forall\lambda \\ P_{coll}(\text{three}) &= P_{ml}(\text{three}|D) = 0 \Rightarrow P(\text{three}|D) = 0 \ \ \forall\lambda \\ P_1(\text{one}|Q_1) &= P(\text{one}|D)P(Q_1|D) = \frac{1}{2}\prod_{q\in Q_1} P(q|D) = \frac{1}{2}P(\text{one}|D) = \frac{1}{2}\frac{1}{2} = \frac{1}{4} \\ P_1(\text{two}|Q_1) &= \frac{1}{4} \\ cs_1(Q_1) &= \frac{1}{4}\log_2 \frac{\frac{1}{4}}{\frac{1}{2}} + \frac{1}{4}\log_2 \frac{\frac{1}{4}}{\frac{1}{2}} = -\frac{1}{2} \end{aligned}$$

For the second query we are in the same situation, because the query has only one term and it is in the vocabulary. Due to the clarity score is calculated over all vocabulary terms, the result is the same.

For the third query we get a zero score because the term is not in the vocabulary (it does not appear in any document). The next query is like the first one but it has also a term which apports nothing. The last query needs different intermediate process but we get the same result:

$$P_1(\text{one}|Q_5) = P(\text{one}|D)P(Q_5|D) = \frac{1}{2}\prod_{q\in Q_5} P(q|D) = \frac{1}{2}P(\text{one}|D)P(\text{two}|D) =$$

$$= \frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8}$$

$$P_1(\text{two}|Q_5) = \frac{1}{8}$$

$$cs_1(Q_5) = \frac{1}{8}\log_2\frac{\frac{1}{8}}{\frac{1}{2}} + \frac{1}{8}\log_2\frac{\frac{1}{8}}{\frac{1}{2}} = -\frac{1}{2}$$

Once we have shown query clarities calculated by Lemur and the ones calculated by us, one question arises: why are not they the same? After some code inspection[7], we have found that Lemur:

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

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

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

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

4.2 Second example

We have expanded the previous example as follows:

$$D_1 = \{\text{one, two, three, four, five}\}, D_2 = \{\text{two, three, four, five}\},$$
$$D_3 = \{\text{three, four, five}\}, D_4 = \{\text{four, five}\}, D_5 = \{\text{five}\},$$
$$Q_1 = \{\text{one}\}, Q_2 = \{\text{two}\}, Q_3 = \{\text{five}\}, Q_4 = \{\text{three, two}\}, Q_5 = \{\text{one, two}\}, Q_6 = \{\text{six}\},$$

Clarity scores calculated by Lemur are:

$$cs(Q_1) = 3.90689, cs(Q_2) = 2.90689, cs(Q_3) = 1.58496,$$
$$cs(Q_4) = 2.61441, cs(Q_5) = 3.40689, cs(Q_6) = 0$$

We can show the calculations for the first query clarity score:

$$P_{coll}(\text{one}) = \frac{1}{15}, P_{coll}(\text{two}) = \frac{2}{15}, P_{coll}(\text{three}) = \frac{3}{15}, P_{coll}(\text{four}) = \frac{4}{15}, P_{coll}(\text{five}) = \frac{5}{15}$$

$$P_{ml}(\text{one}|D_1) = \frac{1}{5}, P_{ml}(\text{one}|D_2) = 0, P_{ml}(\text{one}|D_3) = 0, P_{ml}(\text{one}|D_4) = 0, P_{ml}(\text{one}|D_5) = 0$$

$$P_{ml}(\text{two}|D_1) = \frac{1}{5}, P_{ml}(\text{two}|D_2) = \frac{1}{4}, P_{ml}(\text{two}|D_3) = 0, P_{ml}(\text{two}|D_4) = 0, P_{ml}(\text{two}|D_5) = 0$$

$$P_{ml}(\text{three}|D_1) = \frac{1}{5}, P_{ml}(\text{three}|D_2) = \frac{1}{4}, P_{ml}(\text{three}|D_3) = \frac{1}{3}, P_{ml}(\text{three}|D_4) = 0, P_{ml}(\text{three}|D_5) = 0$$

$$P_{ml}(\text{four}|D_1) = \frac{1}{5}, P_{ml}(\text{four}|D_2) = \frac{1}{4}, P_{ml}(\text{four}|D_3) = \frac{1}{3}, P_{ml}(\text{four}|D_4) = \frac{1}{2}, P_{ml}(\text{four}|D_5) = 0$$

$$P_{ml}(\text{five}|D_1) = \frac{1}{5}, P_{ml}(\text{five}|D_2) = \frac{1}{4}, P_{ml}(\text{five}|D_3) = \frac{1}{3}, P_{ml}(\text{five}|D_4) = \frac{1}{2}, P_{ml}(\text{five}|D_5) = 1$$

$$P(D_i) = \frac{1}{5}$$

$$P(\text{one}|D_1) = 0.14, P(\text{two}|D_1) = 0.02, P(\text{three}|D_1) = 0.02, P(\text{four}|D_1) = 0.02, P(\text{five}|D_1) = 0.02$$

[7] Inside the Lemur project the file to inspect is `SimpleKLRetMethod.cpp`

For the query $Q_1 = \{\text{one}\}$, the set R is only composed by D_1[8]:

$$\begin{aligned}
P(Q_1|D_1) &= 0.14, P(Q_1|D_2) = 0.02, P(Q_1|D_3) = 0.02, \\
&\quad P(Q_1|D_4) = 0.02, P(Q_1|D_5) = 0.02 \\
P_1(\text{one}|Q_1) &= P(\text{one}|D_1)P(Q_1|D_1) = 0.0196, P(\text{two}|Q_1) = 0.0028, P(\text{three}|Q_1) = 0.0028, \\
&\quad P(\text{four}|Q_1) = 0.0028, P(\text{five}|Q_1) = 0.0028 \\
cs_1(Q_1) &= P(\text{one}|Q_1)\log_2 \frac{P(\text{one}|Q_1)}{P_{coll}(\text{one})} + \cdots + P(\text{five}|Q_1)\log_2 \frac{P(\text{five}|Q_1)}{P_{coll}(\text{five})} = \\
&= -0.105
\end{aligned}$$

5 Future work

We have shown a potential application of language models to the analysis of query performance or prediction of query vagueness. During this work, some ideas about its utilisation appeared, and we plan to develop them 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 [25] 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.

6 Conclusion

The language modelling framework has been proved to be a powerful tool to model reality in the field of Information Retrieval. It has been used for ranking firstly, but after that some authors started to use it as a very useful and intuitive query analyser. Since that, language models have grown in complexity, and now they are able to support feedback or personalisation. However, these approaches are only scratching the surface of the possibilities language models offer, and surely there are still several domains to improve by the use of this framework.

One of these areas is the query analysis and its application to predict query performance or query ambiguity. There has been a lot of research done here and the use of language models has proved its convenience, in spite of its complexity.

[8] We have solved this example only for the non-inversion method, because the other implies the calculation of $P(Q_1|D_i)\ \forall i$

In this work we have deeply investigated these topics. Besides that we have tested some available tools and we have found which is the most suitable for continuing developing in this area (Terrier). We have also done calculations by hand to check the results read in the papers and returned by the tools.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] Steve Cronen-Townsend and W. Bruce Croft. Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*, pages 104–109, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[6] 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.

[7] 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.

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

[9] 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.

[10] Djoerd Hiemstra. *Using language models for information retrieval.* PhD thesis, 2000.

[11] 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.

[12] 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.

[13] J. Lafferty and C. Zhai. *Probabilistic Relevance Models Based on Document and Query Generation*, volume 13. Kluwer International Series on Information Retrieval, 2003.

[14] 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.

[15] 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.

[16] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A hidden markov model information retrieval system. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 214–221, New York, NY, USA, 1999. ACM.

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

[18] 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.

[19] 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.

[20] 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.

[21] Xuehua Shen, Bin Tan, and Chengxiang Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2005. ACM Press.

[22] Xuehua Shen, Bin Tan, and Chengxiang Zhai. Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA, 2005. ACM Press.

[23] 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.

[24] 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.

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

[26] 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.

[27] C. Zhai. *Risk Minimization and Language Modeling in Text Retrieval*. PhD thesis, 2002.

[28] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410, New York, NY, USA, 2001. ACM Press.

[29] 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.