Information Retrieval Using Hierarchical Dirichlet Processes

Phil Cowans
Inference Group
Department Of Physics
University Of Cambridge
pjc51@cam.ac.uk

October 26, 2004
Introduction

- This talk is about *ad-hoc* information retrieval.

- In other words, we are given...
  - A collection of documents, $\mathcal{C} = (d_1, d_2, \ldots)$.
  - A query, $q$.

- Our task is to sort the collection in order of *relevance* to $q$.

- The exact definition of relevance is open to interpretation.
Approaches To Information Retrieval

- There are a number of different approaches:
  - Vector space methods
  - ‘Traditional’ probabilistic models
  - Language modelling
    * Uses a statistical language model derived from the query and/or the document.
    * Relevance is defined based on the probability of the query / document under the model, or by comparing models.

- This work extends the language modelling framework.
‘Traditional’ And Vector Space Approaches

- A wide variety of different models, the most successful being BM25.

- Features common to many of the models in this category include:
  - tf.idf like weighting—terms appearing often in the document are more heavily weighted. Terms appearing in many documents are considered less important.
  - Document length normalisation—longer documents are more likely to contain query terms by chance.
Language Modelling Approaches

- Probabilistic models define a probability distribution over the set of all possible texts.

- The majority of methods use *bag of terms* models—The terms in the document are generated independently:

  \[
  \Pr(x) = \prod_{i=1}^{N} \Pr(x_i)
  \]

- Bayes’ theorem can be used to invert the distributions.
Language Modelling Approaches

- There are three main approaches
  - One language model based on the query, used to construct documents.
  - One language model based on each document, used to construct the query.
  - Language models for both the query and the document, relevance defined by comparing the two (KL Divergence)

- Here we train using the documents rather than the queries—more data available.
Smoothing

- Training a language model on a single document/query gives poor performance. Models are smoothed by combining with a collection wide model:

$$ P_s (x \mid d) = \gamma (x, d) P (x \mid d) + (1 - \gamma (x, d)) P_C (x) $$

- Smoothing techniques include Jelineck-Mercer, absolute discounting, and (non-hierarchical) Dirichlet priors.

- $P_C (x)$ is usually either the collection term frequency, or the document frequency. It must be specified ab initio.
The Dirichlet Distribution

- The collection model employed in this work will make use of the hierarchical Dirichlet process. But we’ll begin by introducing a close relative, the Dirichlet distribution.

- The Dirichlet distribution is a probability distribution over probability distributions (conjugate to the multinomial).

- Samples are finite, discrete distributions, \( p = (p_1, p_2, \ldots) \).
The Dirichlet Distribution

- The distribution is given by:

\[
\Pr(p) = \begin{cases} 
\frac{1}{Z(\gamma H)} \left( \prod_{i=1}^{N} p_i^{\gamma H_i - 1} \right) & \text{If } \sum_{i=1}^{N} p_i = 1 \\
0 & \text{Otherwise}
\end{cases}
\]

- \( H = (H_1, H_2, \ldots) \) is a normalised base measure, defining the mean of the distribution.

- \( \gamma \) is a concentration parameter—larger \( \gamma \) values give samples more tightly clustered around the mean.
Draws From A Dirichlet Distribution

Base Measure

\( \gamma = 10 \)

\( \gamma = 1000 \)
Pòlya Urns

- We can sample explicitly from a Dirichlet distribution. Alternatively a sample can be obtained implicitly using a Pòlya urn scheme.

- Samples are obtained by drawing from an urn containing $\gamma H_1$ balls of colour 1, $\gamma H_2$ balls of colour 2 and so on...

- After each sample, the ball is returned, and a new ball is added of the same colour.

- The resulting set of samples are distributed according to a single sample from the Dirichlet distribution.
The Hierarchical Dirichlet Distribution

Each sample location has a label, $y_i$, giving the multinomial from which it is drawn:

$$x_i \sim \text{Multinomial} \left( m_{y_i} \right)$$

$$m_y \sim \text{Dirichlet} \left( \lambda_1 m \right)$$

$$m \sim \text{Dirichlet} \left( \lambda_2 u \right)$$
Oracle Formulation

- The hierarchical version of the Pòlya Urn scheme is the Oracle framework (otherwise known as a Chinese Restaurant Franchise).

- With some probability, new samples are generated using a Pòlya urn local to the related multinomial.

- The remainder of the time, the oracle is asked, which has its own urn.

- The oracle is shared between all multinomials.
Oracle Formulation

\[
\sum_i \frac{n_{i|j}}{n_{i|j} + \lambda_1} \quad \text{Emit } i
\]

\[
\frac{\lambda_1}{\sum_i n_{i|j} + \lambda_1}
\]

\[
\frac{n_i^{(0)}}{\sum_i n_i^{(0)} + \lambda_2} \quad \text{Emit } i
\]

\[
\frac{\lambda_2}{\sum_i n_i^{(0)} + \lambda_2} \quad \frac{1}{N} \quad \text{Oracle}
\]
The Infinite Limit

- The hierarchical Dirichlet process can be viewed as the infinite limit of the hierarchical Dirichlet distribution.

- Importantly, distributions are still discrete, but now over a countably infinite set of states. This allows (approximately) infinite vocabularies to be modelled.

- You can’t sample directly from a hierarchical Dirichlet process, but indirect samples can still be obtained using the oracle formulation.

- (In fact, it makes very little difference whether we use the finite or infinite model, but the infinite model avoids the need to set the vocabulary size).
The Collection Model

- The hierarchical Dirichlet process allows us to specify a generative model of the collection.

- A ‘parent’ distribution over terms is first generated from a Dirichlet process with a uniform base measure and concentration parameter $\lambda_2$.

- A distribution is then created for each document in the collection, using the parent distribution as the base measure. and concentration parameter $\lambda_1$.

- Finally, documents are constructed by drawing terms from the corresponding distribution.
The Collection Model

- This is intuitively appealing, as it is reasonable to assume there is a common distribution (e.g. ‘English’), about which the distributions for individual documents can vary to some extent.

- $\lambda_1$ governs the extent to which document distributions can vary from the base.

- By making the base distribution a random variable, rather than fixing it from the start, information can be exchanged between documents.

- (This is very similar technique to that used in many smoothed $n$-gram language models).
To perform information retrieval, we assume that the query was generated from the same distribution as one of the documents.

Relevance is defined as the probability that the distribution used belonged to the corresponding document:

\[
R(d, q) = \log (\Pr(y_q = y_d \mid x_q, y_C, x_C))
\]

We can use the collection model to find this via Bayes’ rule:

\[
\Pr(y_q = y_d \mid x_q, y_C, x_C) \propto \Pr(x_q \mid y_q = y_d, y_C, x_C) \cdot \Pr(y_q = y_d \mid y_C, x_C)
\]
Prior Distributions

- Note that we need to specify a prior over documents:

\[
Pr(y_q = y_d \mid y_C, x_C)
\]

- In this work the prior is uniform—all documents are \textit{a priori} equally likely to produce the query.

- However, it is possible to specify an arbitrary prior, for example to incorporate additional knowledge about the collection or the user.
An Important Approximation

- Using the oracle formulation is fine if you know how many times the oracle was asked when producing the data we have already seen.

- Unfortunately we don’t know this—we need to marginalise over all possibilities, which is prohibitively expensive.

- To solve this problem, we assume that the oracle was asked the first time that each term is seen in each document, and never asked subsequently.

- (This is essentially the same approximation as ‘update exclusion’ in traditional language modelling).
We make the assumption that the query terms are independent given the collection and the query label.

In other words, we ignore query terms which have been already seen. As the query is typically much shorter than the documents in the collection, this is fairly justified.

The model was implemented using the LEMUR language modelling and information retrieval toolkit.
The Score Function

Putting it all together...

\[ P_I \left( x_q^{(i)} \mid y_q \right) = \frac{tf \left( x_q^{(i)}, y_q \right)}{N_{yd} + \lambda_1} + \frac{\lambda_1 df \left( x_q^{(i)} \right)}{N_{yq} + \lambda_1 \sum_{x'} df \left( x' \right) + \lambda_2} \]

\[ = \frac{1}{N_{yq} + \lambda_1} \left( tf \left( x_q^{(i)}, y_q \right) + \lambda_1 mdf \left( x_q^{(i)} \right) \right) \]

in which the \textit{modified document frequency} is defined as

\[ mdf \left( x \right) \triangleq \frac{df \left( x \right)}{\sum_{x'} df \left( x' \right) + \lambda_2} \]
The Score Function

- Rearranging a bit, and taking logs

\[
\log \left( \Pr \left( x_q^{(i)} \mid y_q \right) \right) = \log \left( \frac{1}{N_{y_q} + \lambda_1} \right) + \log \left( 1 + \frac{\text{tf} \left( x_q^{(i)}, y_q \right)}{\lambda_1 \text{mdf} \left( x_q^{(i)} \right)} \right) + \text{const.}
\]

- Ignoring the constant, and summing over all query terms,

\[
R \left( d, q \right) = \sum_i \log \left( 1 + \frac{\text{tf} \left( x_q^{(i)}, y_d \right)}{\lambda_1 \text{mdf} \left( x_q^{(i)} \right)} \right) + N_q \log \left( \frac{1}{N_{y_d} + \lambda_1} \right)
\]
Interpretation Of Individual Terms

- The individual terms in the score function can easily be interpreted:

\[ \sum_i \log \left( 1 + \frac{tf(x_q^{(i)}, y_d)}{\lambda_1 mdf(x_q^{(i)})} \right) \]  

Logarithmic tf.idf-like term weighting.

\[ N_q \log \left( \frac{1}{N_{y_d} + \lambda_1} \right) \]  

Overall document length normalisation.

- Both of these are commonly found in other methods, and arise naturally from the hierarchical Dirichlet model.

- (Note that this can be regarded as a vector space model with an additional ‘global’ term).
Experimental Tests

- Performance was compared with other methods on TREC-7 and -8 ad-hoc tasks

- (50 queries, 528155 documents, binary relevance judgements)

- Other methods used were:
  - BM-25
  - Twenty-One (Per document language model)
  - KL Divergence (Document and query language models)
  - Hierarchical Dirichlet model
Experimental Tests

- Full query text (title, description and narrative) was used.

- The Dirichlet parameters were set to $\lambda_1 = 1250$ and $\lambda_2 = 750$.

- Preprocessing was limited to:
  - Basic stop word removal
  - Porter stemming
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>TREC-7</th>
<th>TREC-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL-Divergence</td>
<td>21.1%</td>
<td>25.7%</td>
</tr>
<tr>
<td>BM-25</td>
<td>21.5%</td>
<td>24.8%</td>
</tr>
<tr>
<td>Twenty-One</td>
<td>22.2%</td>
<td>26.2%</td>
</tr>
<tr>
<td>Dirichlet</td>
<td>23.3%</td>
<td>27.0%</td>
</tr>
</tbody>
</table>

Average non-interpolated precision over top 1000 documents.
Results

Precision-Recall curves (TREC-7)
Results

Precision-Recall curves (TREC-8)
Further Work

- Relaxing the bag of terms assumption ($n$-gram models).
- Introducing collection structure (paragraph level, multiple collections etc.)
- Avoiding the oracle frequency approximation.
- Mixtures of hierarchies.
- Pitman-Yor processes.
Conclusions

- The hierarchical Dirichlet process can be successfully applied to whole collection modelling for information retrieval.

- By providing a generative model, the assumptions made by the model are made explicit.

- Whilst making minimal assumptions, the model can recover tf.idf like term weighting and document length normalisation.
That’s All...

pjc51@cam.ac.uk

http://www.inference.phy.cam.ac.uk/pjc51/