The five research chapters all report on sets of experiments. Since these experiments share at least some of the same aspects, we introduce the most important aspects of the experimental methodology in this chapter. Besides the general methodology presented, we give per-chapter details when required in the chapters themselves.

We first introduce the evaluation methodology used throughout the thesis in Section 3.2. This section consists of evaluation metrics, significance testing, and test collections and tasks. The second part of this chapter, i.e., Section 3.3, introduces our baseline retrieval model.

3.1 Test Collections and Tasks

One of the main drivers behind successful experimental research in the field of IR is the availability of test collections. These test collections are provided by community efforts like the Text REtrieval Conference (TREC), the Cross-Language Evaluation Forum (CLEF), the INitiative for the Evaluation of XML retrieval (INEX), the NII Test Collection for IR systems project (NTCIR), and the Forum for Information Retrieval Evaluation (FIRE). The main reason for using test collections is that they are reusable, which allows researchers to develop new approaches to a task, assess these on the test collection(s), and compare the results to previous approaches on the same collection(s).

Test collections are constructed after a task has been proposed that needs to be “solved.” These tasks can range from basic ad-hoc retrieval to more complex tasks like list completion or summarization. Once a task is proposed to and accepted by one of the community efforts, a test collection is required to allow researchers to test their systems on this particular task. Test collections usually consist of three parts: (i) the document collection, (ii) a set of test topics, and (iii) assessments for the topics. All three parts depend on the task at hand: for an ad-hoc retrieval, for example, task we would need a collection of web pages, a set of keyword queries, and relevance assessments on a binary level (relevant or not). Other tasks require different collections and topic types, though.

Although test collections are very important for IR research, over the years various authors have shown that problems can arise in the construction and usage of these collections. Here we list three recent papers that discuss separate issues with test collections: (i) assessor expertise [12], (ii) effects of assessor errors [32], and (iii) intra-assessor consistency [165]. Bailey et al. [12] show that different levels of assessor expertise (i.e., topic
creators and experts, topic experts, and non-experts) have low agreement in assessments and that differences in assessments between the various levels affect performance scores of systems. Carterette and Soboroff [32] identify eight assessor models (e.g., fatigued and topic-disgruntled) and use these to simulate assessors for the TREC Million Query Track. They find that different models lead to different system rankings. The models that underestimate the number of relevant documents seem to be more reliable. Finally, Scholer et al. [165] assess intra-assessor consistency by looking at duplicate documents in collections. They find that over 15% of the duplicate documents is assessed inconsistently, indicating that assessment errors not only arise from inter-assessor disagreement, but also from intra-assessor inconsistencies.

Despite the issues that might occur with test collections, we believe the advantages of using these test collections easily outweigh the disadvantages. The issues discussed above, however, serve as reminders of the data with which we are working. Test collections are not flawless and we need to keep this in mind when analyzing the results of our experiments.

In this section we introduce the test collections that we use and the tasks for which they are used. The test topics for all tasks in this thesis follow the standard TREC format, consisting of a title field (a few keywords), a description (a few sentences on what the topic is), and a narrative (a short story on which documents should be considered relevant and which ones should not). For our experiments we are only interested in the title of a topic (i.e., 1–5 term queries), which is comparable to a query submitted to a search engine by an end user. As to relevance assessments, we only use binary assessments: for a given topic, a document is either relevant or not relevant. In case a document is not assessed for the topic, it is considered not relevant.

3.1.1 Blog post collection

The experiments in Chapters 5, 6, and 7 use the TRECBlog06 collection [116], consisting of blog posts collected between December 6, 2005 and February 21, 2006. The collection comes with three document types: (i) feeds (e.g., RSS feeds), (ii) permalinks, and (iii) homepages of the blog. For our experiments, we only use the permalinks, that is, the HTML version of a blog post. During preprocessing, we removed the HTML code and kept only the page title and block level elements longer than 15 words, as detailed in [73]. We remove stopwords but do not apply stemming. In Chapters 6 and 7 we only work with English blog posts and we therefore apply language identification using TextCat.¹ Non-English posts are removed from the collection. The statistics of the collection before and after language detection are listed in Table 3.1.

Blog feed search task The task of blog feed search tests a system’s ability to identify bloggers or blogs that show a recurring interest in a given topic. More details on this task can be found in Chapter 5 on page 57. We use two predefined sets of test topics for this task, which have been created during the TREC Blog track in 2007 and 2008. More details on the topic set, relevance assessments, and characteristics of the queries can be found in Section 5.2.1 on page 63.

¹http://odur.let.rug.nl/~vannoord/TextCat/
3.1. Test Collections and Tasks

<table>
<thead>
<tr>
<th>After boilerplate removal</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of blogs</td>
<td>100,649</td>
</tr>
<tr>
<td>Number of posts</td>
<td>3,215,171</td>
</tr>
<tr>
<td>Index size</td>
<td>12.0 GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>After boilerplate removal and language detection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of blogs</td>
<td>76,358</td>
</tr>
<tr>
<td>Number of posts</td>
<td>2,574,356</td>
</tr>
<tr>
<td>Index size</td>
<td>9.3 GB</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics of the TRECBlog06 collection after preprocessing.

**Blog post retrieval task** When we use a system to perform the blog post retrieval task, we test the system’s ability to return relevant blog posts for a given query. We apply our system to this task in Chapters 6 and 7. The task ran at TREC, as part of the blog track, in 2006–2008 [119, 146, 147]. Each of these TREC editions offers 50 topics and relevance assessments, giving us 150 topics in total. Table 3.2 lists some statistics of the queries in our test collection. We see that more posts were assessed in 2006 than in 2007 and 2008, which leads to more relevant posts per query. As to the number of terms per query, we see that 2008 queries are, on average, quite a bit longer than the 2006 and 2007 queries.

<table>
<thead>
<tr>
<th></th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Assessed posts</td>
<td>67,382</td>
<td>54,621</td>
<td>53,815</td>
</tr>
<tr>
<td>Relevant posts</td>
<td>19,891</td>
<td>12,187</td>
<td>11,735</td>
</tr>
<tr>
<td>Rel. posts/ query</td>
<td>397</td>
<td>244</td>
<td>235</td>
</tr>
<tr>
<td>Query terms</td>
<td>99</td>
<td>85</td>
<td>128</td>
</tr>
<tr>
<td>Terms/ query</td>
<td>2.0</td>
<td>1.7</td>
<td>2.6</td>
</tr>
</tbody>
</table>


3.1.2 Email collection

The test collection that we use in Chapter 8 is the lists part of the W3C collection [192]. This part of the collection comprises 198,394 documents. Not all of these, however, are actual email messages, as some of them are navigational pages. We use a cleaned version of the corpus provided by Gianluca Demartini (with navigational pages removed) and we use thread structure contributed by W3C. After processing the thread structure we end up with 30,299 threads. More details on the corpus are listed in Table 3.3. We remove stopwords, but do not apply stemming. In the same chapter we use an external corpus for query modeling purposes. This corpus is the www part of the W3C corpus, consisting of 45,975 web documents.

2http://ir.nist.gov/w3c/contrib/
3. Experimental Methodology

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of emails</td>
<td>174,299</td>
</tr>
<tr>
<td>Average email length</td>
<td>327</td>
</tr>
<tr>
<td>Average email length (w/o quotes)</td>
<td>234</td>
</tr>
<tr>
<td>Number of threads</td>
<td>30,299</td>
</tr>
<tr>
<td>Average thread length</td>
<td>687</td>
</tr>
<tr>
<td>Average number of emails</td>
<td>3.87</td>
</tr>
<tr>
<td>Maximum number of emails</td>
<td>116</td>
</tr>
</tbody>
</table>

Table 3.3: Corpus characteristics of W3C lists.

Email finding topics  We use the topic sets developed for the Discussion Search (DS) task as TREC [38, 177]: 59 topics from 2005 and 50 topics from 2006. Relevance assessments for the discussion search task come on multiple levels, but for our experiments we focus on the topical relevance of emails, resulting in binary assessments.

3.2 Evaluation

An important aspect of our methodology is measuring. In this section we first introduce the various tasks and test collections that are used in this thesis. We also discuss the metrics we use to assess performance of our models and the significance testing we perform to compare results.

3.2.1 Evaluation metrics

To measure the effectiveness of our models in Chapters 5–8, we use a set of common IR metrics [126]. We can divide the IR metrics into metrics that are (i) recall-oriented and (ii) precision-oriented. Recall-oriented metrics measure how well a system is able to retrieve all relevant documents that exist, whereas precision-oriented metrics measure how many documents within the retrieved set of documents are relevant. Here, we briefly explain the four metrics we report on in this thesis: mean average precision (MAP), mean reciprocal rank (MRR), and precision at ranks 5 and 10 (P5, P10).

Various other metrics have been proposed besides the metrics we discuss below. Most notably, Yilmaz and Aslam [211] introduce infAP as a “replacement” for average precision. The motivation behind infAP is similar to that of bpref, introduced by Buckley and Voorhees [28], in that they both assume the relevance judgments to be incomplete. Compared to MAP these metrics are more stable, however, they result in similar system rankings. Since we mostly look to compare various methods (systems) in this thesis we use MAP as our metric. One advantage of using MAP is that results in the various chapters of this thesis are easily comparable to results in previously published papers, since MAP is (still) the most commonly used metric.

Mean average precision (MAP)  This is a recall-oriented metric used most commonly in research in the field of IR. For each relevant document in the returned document list we take the precision at the position of that document. We sum over these precision
values and divide it by the total number of relevant documents. This gives us the average precision (AP) for a query. When we take the mean of AP values over a set of test queries, we get the mean average precision (MAP) for a system on that set of queries.

\[
AP = \frac{\sum_{r=1}^{N} P(r) \cdot rel(r)}{|R|},
\]

where \(R\) is the set of relevant documents for a given query, \(r\) is the position in the ranked list, and \(N\) is the number of returned documents (in most TREC tasks \(N = 1,000\)).

We then calculate the precision at rank \(r\):

\[
P(r) = \frac{\sum_{t=1}^{r} rel(t)}{r},
\]

and finally we need a binary function that indicates whether or not the document at rank \(r\) is relevant:

\[
rel(r) = \begin{cases} 
1 & \text{if } r \in R \\
0 & \text{otherwise.}
\end{cases}
\]

As mentioned before, we can average the AP values to obtain the MAP of a system.

**Precision at rank \(r\) (Pr)** The precision at rank \(r\) metrics (P5 and P10) are calculated in the same way as Equation 3.2 and indicate the percentage of relevant documents within the top \(r\) returned documents. In web search related tasks this metric is often considered important, because users tend to look only at the top 10 results of a ranked list.

**Mean reciprocal rank (MRR)** The final precision-oriented metric we report on is the mean reciprocal rank. This metric indicates how good a system is in returning the first relevant document as high up the ranking as possible. To measure this we take the reciprocal of the position of the first relevant document. When a system returns a relevant document on the first position, its reciprocal rank (RR) is 1, but when the first relevant document is returned on position 8, we get an RR of 0.125. After taking the average over the RR values of a set of queries we get the mean reciprocal rank for a system on that set of queries.

### 3.2.2 Significance testing

In Chapters 5–8 we introduce approaches that should improve performance on the tasks in these chapters. To test if our proposed approaches really do show improvements we compare their scores to baseline scores. These baseline scores indicate how the system performs without our approach. When comparing two runs, we want to test for significant differences between them. To this end we use a two-tailed paired t-test. Smucker et al. [176] show that in practice there is no difference between the t-test and the randomization test, although the latter is a more principled choice. In this thesis we opt for the t-test, however, given its simplicity and commonness in IR papers.

In our result tables we show significant differences for \(\alpha = .01\) and \(\alpha = .05\), the former being stronger than the latter. Results marked by ✶ and ✷ reflect significant improvements or drops for \(\alpha = .01\) and ▲ and ▼ do the same for \(\alpha = .05\).
3. Experimental Methodology

3.3 Baseline Retrieval Model

In Chapters 6–8 we use the same baseline retrieval model on which we build our improvements. As our baseline system we use a language modeling approach to IR [39]. Working in the setting of generative language model, one usually assumes that a document’s relevance is correlated with query likelihood [72, 133, 149]. Within the language modeling approach, one builds a language model from each document, and ranks documents based on the probability of the document model generating the query, that is \( P(D|Q) \). Instead of calculating this probability directly, we apply Bayes’ Theorem and rewrite it to

\[
P(D|Q) = \frac{P(Q|D)P(D)}{P(Q)}.
\]

(3.4)

The probability of the query \( P(Q) \) can be ignored for the purpose of ranking documents for query \( Q \), since it will be the same for all documents. This leaves us with

\[
P(D|Q) \propto P(D)P(Q|D).
\]

(3.5)

Assuming that query terms are independent from each other, \( P(Q|D) \) is estimated by taking the product over each term \( t \) in query \( Q \), resulting in

\[
P(D|Q) \propto P(D) \prod_{t \in Q} P(t|D)^{n(t,Q)}.
\]

(3.6)

Here, \( n(t, Q) \) is the number of times term \( t \) is present in the query \( Q \). To prevent numerical underflows, we perform the computation in the log domain (thus compute the log-likelihood of the document being relevant to the query). This leads to the following equation:

\[
\log P(D|Q) \propto \log P(D) + \sum_{t \in Q} n(t, Q) \log P(t|D).
\]

(3.7)

Finally, we generalize \( n(t, Q) \) so that it can take not only integer but real values. This will allow more flexible weighting of query terms. We replace \( n(t, Q) \) with \( P(t|\theta_Q) \), which can be interpreted as the weight of the term \( t \) in query \( Q \). We will refer to \( \theta_Q \) as the query model. We also generalize \( P(t|D) \) to a document model, \( P(t|\theta_D) \), and arrive at our final formula for ranking documents:

\[
\log P(D|Q) \propto \log P(D) + \sum_{t \in Q} P(t|\theta_Q) \log P(t|\theta_D)
\]

(3.8)

Here, we see the prior probability of a document being relevant, \( P(D) \) (which is independent of the query \( Q \)), the probability of observing the term \( t \) given the document model, \( \theta_D \), and the probability of a term \( t \) for a given query model, \( \theta_Q \).

In Chapter 5 we build on the language modeling approach for IR, but we adjust the model to fit the task of finding bloggers (viz. Section 5.1). Chapter 6 only uses this baseline model and does not change anything to it. In Chapter 7 we focus on improving the estimate of \( P(t|\theta_Q) \) and in Chapter 8 we focus on that part again, but also on \( P(D) \), the prior probability of the document.