The Expected Metric Principle for Probabilistic Information Retrieval

by

Harr Chen

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2007

Author .....
Department of Electrical Engineering and Computer Science
January 19, 2007

Certified by .....
David R. Karger
Professor of Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

The Expected Metric Principle for Probabilistic Information Retrieval

by

Harr Chen

Submitted to the Department of Electrical Engineering and Computer Science
on January 19, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Traditionally, information retrieval systems aim to maximize the number of relevant documents returned to a user within some window of the top. For that goal, the *Probability Ranking Principle*, which ranks documents in decreasing order of probability of relevance, is provably optimal. However, there are many scenarios in which that ranking does not optimize for the user's information need. One example is when the user would be satisfied with *some* limited number of relevant documents, rather than needing *all* relevant documents. We show that in such a scenario, an attempt to return *many* relevant documents can actually reduce the chances of finding *any* relevant documents.

In this thesis, we introduce the *Expected Metric Principle*, which generalizes the Probability Ranking Principle in a way that intimately connects the evaluation metric and the retrieval model. We observe that given a probabilistic model of relevance, it is appropriate to rank so as to directly optimize these metrics in expectation. We consider a number of metrics from the literature, such as the rank of the first relevant result, the %no metric that penalizes a system only for retrieving *no* relevant results near the top, and the diversity of retrieved results when queries have multiple interpretations, as well as introducing our own new metrics. While direct optimization of a metric's expected value may be computationally intractable, we explore heuristic search approaches, and show that a simple approximate greedy optimization algorithm produces rankings for TREC queries that outperform the standard approach based on the probability ranking principle.

Thesis Supervisor: David R. Karger
Title: Professor of Computer Science

Acknowledgments

I am literally writing these acknowledgments an hour before the thesis deadline, so this section will necessarily be brief.

David R. Karger, my advisor.

My friends. You know who you are.

My current and former officemates.

The other students, staff, and faculty of CSAIL.

The Office of Naval Research, who supports me through a National Defense Science and Engineering Graduate Fellowship.

Most importantly, my parents Baokai Chen and Ping Zhu. I owe my entire life to you, mom and dad.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we propose a novel approach to probabilistic information retrieval (IR). Our core contribution is to observe that a retrieval system's quality evaluation measures (its metrics) directly motivate a natural ranking objective. Building on a standard probabilistic model of text in documents, and a variety of evaluation metrics (both previously introduced in the literature and our own), we generalize the standard ranking principle to explicitly incorporate the metric in the model.

1.1 Motivation

Mary wants information related to a song she heard on the radio. Ideally, she wants as much information on the song as possible - for example, information about its history and artist, the song lyrics, and a website for purchasing the song would all be relevant to her. Mary enters the song title into an Internet search engine. In present Internet search engines, the top ten result set overwhelmingly consists of the lyrics listings hosted on different websites, without any other song information. Any one of the lyrics documents is useful to Mary. However, the result set of ten lyrics documents taken together is not as useful as a result set that includes a diverse mix of lyrics pages and other song-related information. The reason that search engines return the homogeneous result set is that standard ranking principles judge the relevance of each document individually, without relation to the other returned documents. In

turn, many standard evaluation metrics judge a result set's quality on the *number* of relevant results, which in this case would misidentify the homogeneous result set as being of high quality. What if our metric instead focused on finding *one* right answer? In such a case, that one right answer may be the lyrics, or a music download site, or artist information, and so a retrieval engine that aims to find one right answer, should also find a diverse set of answers. In this thesis, we define a new metric that captures the notion of getting one relevant document, and show how it can be naturally incorporated into the retrieval algorithm.

The independent relevance assumption is also problematic when considered from the perspective of multiple users. Consider another example that involves two hypothetical users of an Internet search engine, Bob and Jane. Bob is interested in finding information about computer viruses, whereas Jane wants information on biological viruses. They both enter "virus" as the query to an Internet search engine. A typical search engine would overwhelmingly favor one interpretation of "virus" over the other - for example, Google's search engine returns as the top ten results ten different sites that are all very relevant to computer viruses. Individually, each site is relevant to Bob, so Bob would be satisfied with any of these ten results. However, none of the results satisfies Jane's information need. We would rather satisfy both users, but each individual result ends up favoring the same user interpretation. Standard metrics do not capture these multiple perspectives well—but there exist metrics that specifically reward for retrieving different interpretations of a query. A retrieval engine should be able to directly optimize for these multiple interpretations, and we will show how this is done by incorporating the diversity metric into the retrieval model.

To reiterate, the general principle is that the metric informs the model. Before proceeding, we first take a step back and define our problem space more carefully.

1.2 Problem Setup

Information retrieval (IR) is, broadly speaking, the art and science of building systems that assist human end users in finding information. More specifically, *text retrieval*

looks at the problem of finding information in text, which is of critical importance in areas such as World Wide Web search and digital libraries. Search engines such as Google are centered around the text retrieval problem. One can imagine that without search functionality, the web would be a much more difficult place to navigate.

In the canonical retrieval setup, information is stored in a *corpus* (collection) of *documents*. The corpus could be, for example a flat collection, a hierarchical collection, or relational database. The documents could be text, multimedia content, or other forms of data.

For example, in the web domain, webpages are documents, and page links and domain hierarchies define some notion of structure on the collection.

We will be looking specifically at the text retrieval problem over a collection of documents. Such a system is evaluated on the basis of the *relevance* of its output to the user's input. In the standard approach, seen in most web search engines, the user enters a textual query, and receives a ranked list of documents, called the *result set.*[1]

We take a *modelling* approach to the problem—by defining a *model* that assigns numerical "scores" to various possible outcomes (i.e., result sets) that the IR system could produce, we could select an "optimal" outcome to present to the user, with respect to the model. In contrast, *ad hoc* approaches encompass developing retrieval methods in a heuristic manner, by incrementally applying techniques that are not necessarily theoretically well-founded, but are shown to give good empirical performance.

In general, modelling approaches allow us explicitly clarify our assumptions. Most importantly, models provide a framework that allow us to explore how principled changes in our assumptions can be easily incorporated. It is such a change that forms the basis for our work in this thesis.

However, ad hoc approaches have the potential of better empirical performance because they are specifically tuned on performance. Of course, with these approaches one runs the risk of finding heuristics that do not generalize well, so extensive exper-

[1]Note the mathematically inaccurate usage of the term set—while the result documents are each unique, we do impose an ordering on them.

imentation is generally necessary.

IR systems are also rigorously evaluated, generally by using standardized collections of test queries and corpora. The queries are annotated by humans with relevance judgments on documents in the corpora. In this sense, we have a set of "right answers" to the queries. To test an IR system, we then run it on the test queries, and compare the result sets produced by the IR system to the previously annotated relevance judgments. This comparison is captured by the notion of a *metric*, a numerical score of the "goodness" of the IR system's results. Different metrics capture different notions of this goodness, some of which trade off against each other. For example, some metrics may measure the total number of relevant documents returned, whereas others only care about the rank of the first relevant document returned.

In this thesis, we will look at one step toward unifying the model-based and empirical approaches, by directly feeding in the *metric* technique as an input to a well-defined *model*. Given that metrics are themselves rigorously defined, we can directly use the *expected value* of the metrics as an objective function in a *probabilistic* model. We contrast this with the standard method of ranking using probabilistic models, the *Probability Ranking Principle* (PRP). We then tackle the computational issues associated with our new objective functions, many of which are intractable to exactly optimize.

1.3 Retrieving for Diversity

To make these general notions concrete, we use the case of retrieving for a *diverse* set of results as a motivating example throughout the thesis. Traditionally, IR systems try to return as many relevant documents as possible. However, that is not the only possible goal. For example, since a single relevant result often provides "the answer" to the user's query, we might be concerned only with whether our system returns *any* relevant results near the top. This is plausible for question answering, or for finding a homepage. It also captures a notion of "bare minimum success" that can be meaningful for hard queries.

In such a case, we would rather the result set be *diverse*, as we are more likely to satisfy a broad range of interpretations in that case. We use some previously defined metrics in the literature, as well define a new class of metrics which capture the diversity/focus tradeoff in a result set particularly well. We show how our general approach can be applied to these metrics, and demonstrate compelling experimental results on standard IR corpora.

Intriguingly, while explicitly aiming only to find one relevant document, we demonstrate the unplanned effect of increasing the diversity of documents at the top. This highlights one way in which seeking *one* relevant document is different from seeking *many*. If a query has multiple interpretations, or if there are multiple *instances*, it may be hard to decide which is the proper one. For example, "trojan horse" can refer to a Troy war construct, a malicious computer program, an antique store, a board game, etc. PRP ranking puts all its eggs in one basket—it identifies the most likely interpretation, and finds many results for that one. But an algorithm that needs only one relevant document can do better by retrieving one document for each case, thus satisfying the goal *whichever* interpretation or instance is desired.

1.4 Thesis Overview

In this section, we summarize the major components of this thesis.

1.4.1 Probabilistic Models (Chapters 2 and 3)

One way of viewing the information retrieval problem is from a *probabilistic* perspective. Under such an approach, ranking is performed according to a probability value. The standard *probability ranking principle* (PRP) ranks by the independent probability of relevance of each corpus document. By applying Bayes' Rule, PRP is equivalent to ranking by a ratio—the probability that the document is *generated* by the *relevant document distribution* to the probability that the document is generated by the *irrelevant document distribution.* The relevant and irrelevant document distributions are probability models that simulate the process of how documents are created, or

generated. Different documents have different probabilities of being generated from either distribution; in particular, documents likely to be relevant should have a higher probability of being generated from the relevant distribution. In practice, this means we use the query or other related documents as *training data* that allows us to ascertain *parameter estimates* for each distribution. In other words, training adjusts the distributions to favor documents similar to the training data, as we would expect—documents relevant to a query should contain the query terms, and in a sense be "similar" documents.

In this thesis, we will be using *multinomial* probability distributions to represent the process of document generation, and *Maximum a Posteriori* estimation with a *Dirichlet* prior to set the specific parameters of the distributions. A multinomial distribution is similar to a many-faced, unfair die; a document in this model is viewed as a series of rolls of die, with each face representing a possible term. Obviously, each term can have different probabilities—any document is much more likely to have the word "it" than "borborygmus."[2] A Dirichlet prior allows us to specify preexisting beliefs about the likelihoods of terms and is used to normalize and smooth the probability distributions.

1.4.2 Evaluation Metrics (Chapter 4)

Information retrieval systems are evaluated on the basis of standardized data, by using numerical measures called *metrics*. The standard way of performing evaluation is to run the same set of queries, over the same corpus, using multiple retrieval systems. For each retrieval engine, the result sets from each query are scored according to the metric and a relevance standard. This relevance standard is usually annotated corpus/query relevance information as judged by a human expert. The individual result set scores are then aggregated by taking an average (usually an arithmetic mean, but some metrics also use other means). For example, a commonly used metric is precision at ten—each result set is scored on the basis of how many relevant documents (relevant

[2]Borborygmus is the growling sound the digestive system makes as gas moves through the intestines.

as determined by a human annotator) appear in the top ten, divided by ten. The total precision at ten is simply the arithmetic mean of the precision scores over all of the queries.

Different metrics measure result set quality in different ways. The common *precision and recall* metrics both reward systems that return many relevant results. However, there are circumstances when we would rather emphasize finding a result early, or finding a relevant result at all. The former case is covered by metrics such as *reciprocal rank* and *search length*, both of which are computed on the basis of the *first* relevant result's rank. The latter is covered by a new class of metrics we introduce called *k-call at rank n*, which rewards systems for having at least k relevant results in the top n results.

Other metrics look at still more complex notions of relevance—for example, *mean average precision* is a variant of precision that weights relevant results found early in the result set more highly than later results. There are also metrics that look at relevance in a multifaceted manner, such as *instance recall*, which is computed on the basis of multiple annotators' relevance annotations. Clearly, the wide variety of metrics indicate that there are many different ways of looking at the quality of a result set. These measures are not necessarily correlated or even related at all. As previously introduced, the probabilistic model ranks in a straightforward manner by the probability of relevance, which cannot simultaneously be optimal for all of these metrics. This motivation is the basis for our work.

1.4.3 Synergizing Metrics and Models (Chapters 5 and 6)

Our primary contribution is in drawing a connection between the probabilistic model and the evaluation metric. The standard ranking principle, the probability ranking principle, orders the result set by the probability of relevance of each document. We generalize PRP to the *Expected Metric Principle* (EMP), which chooses the best result set (as a whole) by computing a score for each possible result set; this score is the *expected value* of a given metric computed on the result set under the probabilistic model. The EMP then states that the best result set is the one with the best (highest

or lowest) score. For example, if we applied the EMP approach to the 1-call at rank ten metric (indicating that we want a result set of size ten), we would compute the expected value of the 1-call metric for each possible set of ten documents from the corpus, and choose the result set with the highest such score. In chapter 5, we apply EMP to a number of IR metrics. We confirm the previously known result that PRP is optimal for precision and recall, as well as finding that certain variants of the two are also optimized by PRP under EMP. For other metrics such as reciprocal rank and search length, as well as our new k-call, the EMP suggests novel ranking formulae.

For any result set of a reasonable size, it would be very computationally expensive to compute a score for each possible result set. Hence, practically implementing the EMP requires *algorithmic heuristics* for tractability. A straightforward *greedy algorithm*, which selects each document of the result set successively in rank order, is a practical approach that reduces the combinatorially explosive number of results sets that need to be evaluated, to a polynomial number. In the greedy algorithm, we select each document in turn assuming the previously selected documents are fixed; this one degree of freedom allows us to locally optimize for the metric at each rank fairly easily. Another approach that can be used in conjunction with a greedy approach is *pruning*, where the corpus size is effectively reduced by first running a simple ranking procedure (such as PRP) and considering only the top documents from the first-cut ranking for the full EMP ranking. In chapter 6, we derive simplified ranking algorithms for many of the metrics under the greedy algorithm. Interestingly, 1-call, search length, reciprocal rank, and instance recall end up naturally optimized by the same greedy approach, and so we can get "four-in-one" with the same greedy algorithm.

One could argue that it is "cheating" to optimize for a specific metric. Such an argument, however, is more properly a commentary on the validity of the metric and not the principle. Given a metric that properly attributes success to the "right" results sets, it is only natural to directly aim to optimize the metric. As a comparison, in some other fields such as computer performance testing, manufacturers have been accused of tuning their software to work well on specific third party benchmark software. In

this case, however, the accusation is more appropriately that the benchmarks are not representative of real world usage. By analogy, if the IR evaluation metric is not representative of what makes a good system, then the metric should be changed.

1.4.4 Empirical Results (Chapter 7)

To verify our approach, we also perform experiments on standardized TREC data. We find consistent performance improvements on each metric when we use an EMP ranker tuned to the metric, compared to a PRP baseline. Generally, these improvements are statistically significant unless the baseline is already performing very well, in which case further incremental improvements are possible but not as strong.

Looking specifically at the problem of diversity, we give anecdotal evidence that the 1-call greedy algorithm promotes diversity by looking at ambiguous queries on the Google search engine. We observe that while the probability ranking principle tends to return documents only relevant to the "majority vote" meaning of the query, our approach satisfies that meaning but simultaneously returns results relevant to other, rarer meanings of the query. We follow with more quantitative evidence based on TREC results with multiple raters, where our approach satisfies more raters than PRP, and TREC results with subtopic annotations, where our approach retrieves more subtopics than PRP.

1.5 Related Work

Throughout the body of this thesis, we will call out related work as appropriate for the context.

Our discussion of related work splits into three categories: definitions of and motivations for retrieval metrics, algorithms for optimizing those metrics, and approaches to diversity in result sets.

1.5.1 Beyond Precision and Recall

The 1-call metric we propose is essentially one minus the %no metric, which is studied by Voorhees [27]. She finds that the metric was less stable than traditional measures. However, this instability does not affect our ability to probabilistically model and optimize for it.

Cooper [6], who introduces the search length metric, argues that trying to retrieve as many documents as possible is not necessarily the appropriate objective for meeting user information need. Cooper explicitly divides an information request into a "relevance description" (i.e., a query) and a *quantification* that specifies the desired number of relevant results. He defines a class of search length metrics, which measure the number of irrelevant documents a user would have to examine before finding a "sufficient" number of relevant documents. Cooper notes that this sufficiency may be met with one relevant document, with a fixed number k of relevant documents, with a proportion of all the relevant documents, with the entire set of relevant documents, or some combination thereof. Our paper focuses on the case of "one document sufficiency," though we also touch on the "k document sufficiency" case when we define k-call later in the paper.

Gordon and Lenk [12, 13] demonstrate that PRP is optimal only under a specific set of circumstances: the IR system accurately predicts probability of relevance, those probabilities are certain, and documents are judged relevant independent of other retrieved documents. They show that when these conditions are met, a signal detection decision-theoretic approach and a utility theoretic approach both justify PRP. However, when the third condition is not met—that is, when documents are not independently relevant—they show that the expected utility of a PRP-ranked result set could be suboptimal. Because in reality the relevance of documents is dependent on the other retrieved documents, this finding further indicates that PRP is not always an appropriate choice.

Cronen-Townsend, *et al.* [8] introduce the *clarity* measure in an attempt to determine whether a user had formulated a good query. Clarity measures the difference

in distribution between words in a query result set and words in the corpus. The authors argue that a high clarity, reflecting a returned document distribution quite different from that of the corpus, indicates that the user has succeeded in specifying a good query, one that returns a distinctive type of document. One might argue from the perspective of diversity that a high clarity is indicative of a result set that has failed to hedge, and that is therefore more likely to have no documents relevant to a user's particular sense of the query.

Shah and Croft [23] explore the problem of *high accuracy retrieval*, where the objective is to have high precision in the top document ranks. They argue that *mean reciprocal rank* is a useful metric for this scenario. As previously mentioned, we also demonstrate the applicability of our heuristics to MRR.

The event of returning zero relevant documents, whose probability we try to minimize in the $k = 1$ version of our objective function, is precisely the event that the TREC Robust Retrieval Track [28] set out to avoid. Our experiments show that we experience this "abject failure" on the robust track less than a PRP baseline.

1.5.2 Algorithms

Our approach fits within a general *risk minimization* framework propounded by Zhai and Lafferty [30]. They observed that one could define an arbitrary numeric *loss function* over possible returned documents rankings, which measures how unhappy the user is with that set. The loss function generally depends on unavailable knowledge about the relevance of particular documents. But given a probabilistic model, one can compute an *expected value* for the loss function, or *expected loss*, and return a result that optimizes the expected loss. Much of our paper deals with the loss function that is (say) -1 when the top ten results contain a relevant document (indicating a positive satisfaction) and 0 when it does not.

Gao, *et al.* [10] introduce a linear discriminant model for retrieval, and show that training the parameters of the model to empirically optimize for average precision over the training set is preferable to optimizing for likelihood. We are also arguing for optimizing for the metric, but we derive the optimization from a probabilistic

model, rather than from a training set of queries and document judgments.

Bookstein [3] proposes a sequential learning retrieval system that bears some similarity to ours. He argues that a retrieval system should sequentially select documents according to the probability of relevance conditioned on the selection and relevance of previous documents (essentially relevance feedback). However, his procedure requires explicit user feedback after each result retrieved, whereas our system proposes an objective function and then uses a sequential document selection algorithm to heuristically optimize that objective without further user input.

Our greedy algorithm for achieving perfect precision seems related to *pseudo-relevance feedback*, an approach commonly used in the literature to improve overall retrieval performance on standard metrics [4, 9]. Our metric for retrieving at least one relevant document, on the other hand, produces an algorithm that appears to be doing *negative* pseudo-relevance feedback. In either case, rather than feeding back all of the top documents at once, we progressively feed back more and more top relevant documents in selecting latter-ranked documents.

1.5.3 Diversity

Recent work [5, 29] has developed heuristics for increasing diversity, but our approach appears to be the first in which diversity arises automatically as a consequence of the objective function rather than being manually optimized as a proxy for the true objective of interest. As a benefit, there are no new "parameter knobs," beyond those already used in probabilistic document models, that must be tweaked in order to tune our algorithms.

In their subtopic retrieval work, Zhai, *et al.* [29] posit, as we do, that there may be more than one meaningful interpretation of a query. They assume that a query may have different subtopic interpretations, and reorder the results so that the top includes some results from each subtopic. Their system involves separate consideration of *novelty* and *redundancy* in a result set, which are then combined via a cost function. Our approach, in contrast, aims directly at the goal of maximizing the chances that the user will get an answer to "their" interpretation of the query. Aiming directly

arguably is beneficial in that it reduces the number of system elements, such as novelty and redundancy, whose interactions we have to design and tweak. Conversely, it is possible that by modeling novelty and redundancy richly, the Zhai, *et al.*— model can outperform our simpler one.

The work of Zhai, *et al.* is in turn based on Carbonell and Goldstein [5]'s *maximum marginal relevance* (MMR) ranking function. They argue for the value of diversity or "relevant novelty" in the results of a query, and propose MMR as an objective that introduces such diversity in a ranked result set. Our greedy heuristic for optimizing the "one relevant document" objective simplifies to a computation that bears some relation to the MMR computation. However, MMR is advanced as a heuristic *algorithm* for reducing redundancy and achieving the hard-to-define notion of *diversity*, which in turn is believed to be *related* to the desired objective. Our ranking algorithm arises naturally from the application of a simple greedy heuristic to the optimization of a clear, natural, formally defined objective function. In addition, while the iterative greedy approach is implicit in the definition of MMR, our greedy approach is simply one heuristic applied to optimizing our well-defined objective function; we expect that better optimization algorithms such as local search would yield improved values for our objective, which should translate into improved retrieval performance.

Our goal of retrieving one relevant document, and its inherent diversifying tendency, bears superficial similarity to clustering, in the sense that clustering is also used as an approach to quickly cover a diverse range of query interpretations [19]. Our technique sidesteps the need for clustering interface machinery, utilizing the standard ranked list of documents instead. Furthermore, we aim to directly optimize the probability that a user finds a relevant document, rather than going through the intermediate notion of separate document clusters. Again, this avoids the need to define and tweak new algorithmic parameters.

Chapter 2

Retrieval Basics

In this chapter we will present the basic retrieval pipeline. We will go over some standard textual preprocessing methods, a basic probabilistic algorithm for selecting and ranking documents from a collection, and some query reformulation techniques. We will defer our discussion of IR system evaluation until chapter 4, after our discussion of retrieval methodology. The discussion here parallels that of many introductory IR books. See for example chapters 2, 5, 7, and 8 of Baeza-Yates and Ribeiro-Neto's book for a more in-depth discussion of retrieval models, query reformulation, text processing, and index building, respectively [1]. In this chapter, we have chosen to focus on only one class of retrieval models, the probabilistic models. For a brief overview of the two other major classes of models, boolean and vector space, we refer the reader to appendix B.

The standard modelling approaches share a great deal of similarity. They rely on the fundamental atoms of *terms* and *documents*. A term is any word or other indivisible unit of language (for example, a phrase) that appears in the corpus, after it has been preprocessed. In this thesis we will generally use "term" and "word" interchangeably because we do not consider phrase-level terms. A document then is a multiset of terms (referred to as a *bag of words*), or equivalently, a list of its terms along with their frequencies. Note that the models that we will be examining disregard word order within documents—this is a commonly made IR assumption, though more sophisticated models could potentially account for them. When the user

wants to perform a search, he or she enters a *query*, which is preprocessed in the same way as the documents, and used as another bag of words. The magic of the specific IR model is the *selection* and *ranking* of documents from the entire document set. Each model ultimately outputs either a selected subset of the corpus documents as the result set, usually (though not always) with a rank ordering of those results.

2.1 Text Preprocessing

Before we can perform any actual retrieval, we must have some internal representation of the documents and queries that are more directly usable than flat text strings. The first step in shaping the corpora and queries is to normalize the text.

2.1.1 Just the Words, Ma'am

Typically, text is full of rich linguistic structure. This thesis, for example, has punctuation, capitalization, spacing, and other information that makes it human-readable. In our work, we have chosen to ignore this structure. While it is potentially a rich source of retrieval guidance, it greatly convolutes the retrieval models that we will be examining. Hence we remove or normalize out this kind of structure before further processing, leaving just the raw words. It is at this stage that we could also lose word ordering, and essentially jumble the document terms together.

Sometimes our data may also include other forms of structure that we would need to strip out. For example, if our documents were HTML webpages, we may have to remove tags, or use heuristics to incorporate information from the tags as metadata. Since users do not see the tags and thus do not search for them by name, they would not be useful words to index by directly. Of course, the tags can inform the search in other ways; this direction is an actively explored area of research.

2.1.2 Stopwords

Most text includes many words that occur with high frequency, but are relatively meaningless as content words. In English, words such as articles ("a", "the"), connectors ("and", "or"), etc., tend to skew the distribution of words and inflate the index and word counts. Other languages usually have analogous such words. Furthermore, these common words are also generally devoid of information—that is, knowing how many times "a" or "and" shows up in a document does not help us. In fact, all they do is introduce noise to our word tallying, and make the entire index larger and slower to use.

The simplest way to overcome this issue is to strip away these words entirely. We call these words *stopwords*—appendix D gives a sample listing of stopwords (in fact, the list that we filtered on for our experiments).

Removing stopwords is not always productive—for example, "the" and "who" are often considered stopwords, so a search on "the who" would be vacuous if those words were already stripped out. Thus, some search systems do not filter on stopwords at all. In general, however, filtering with stopwords is a commonly used and accepted textual preprocessing heuristic.

2.1.3 Stemming

If a user searches on a word such as "implementation," we would presumably like to match that against words such as "implemented", "implements", "implement", etc. These are effectively the same word, but in different parts of speech. A naïve approach would consider them to be separate terms, but we can match them if we first *stem* all the words in our corpus—that is, cutting out prefixes and suffixes (together known as affixes).

There are a variety of automated stemming algorithms that one could use. One of the most common, and the one we use, is the *Porter stemming algorithm*, which we briefly describe here.

A Porter stemmer maintains a list of grammar-like rules that allow us to transform

certain word suffix strings to shorter forms. The stemmer repeatedly applies these rules until no rules can be further applied. If the current suffix matches multiple rules, we apply the rule that matches the longest suffix. The resulting stemmed word is often not exactly the linguistic root (that is, the morphological root) of the word, but can be seen instead as a label for an equivalence class of all words which stem to the same core string.

As an example, two rules that are used is $s \Rightarrow \emptyset$ and $xes \Rightarrow x$. Given just these two rules, the word "cats" would stem to "cat", and the word "taxes" would stem to "tax" (rather than "taxe", since we always apply the rule that matches the longest suffix).

2.2 Index Building

Now that we have the text in a clean form, we can proceed to build a lookup structure over the text so that our ranking procedures can be done efficiently. This structure is called the *index*. The simplest kind of index facilitates sequential scanning through the corpus documents, by mapping document identifiers with the terms (and possibly also their frequencies and/or positions) that occur in those documents. This is really no different from sequentially scanning through the flat raw text for every search, except that we may be able to store the document data more compactly, and depending on the exact index structure, we may be able to facilitate random access to documents by identifier.

Another commonly used supplemental data structure is the *reverse index*, which is essentially a performance optimization. When queries are short and discriminative, it is unlikely that many documents will contain the query terms—most documents will automatically be out of consideration for that reason. A reverse index facilitates fast elimination of documents by associating index terms with documents that they appear in.

The terms that make it into the index are called, appropriately enough, *index terms*, and may be a further refinement of the terms that come out of text prepro-

cessing. Specifically, sometimes we may have limits as to the number of index terms, or a threshold of term occurrence before we add it to the index at all. This type of pruning is generally done at the index-building stage.

2.3 Probabilistic Models

Now that we have built up an index, we now turn to the actual algorithmic approaches to ranking the documents. There are three broad categories of "classical" retrieval approaches: *Boolean* models that represent queries and documents as logical statements and find which documents "satisfy" the query, *vector space* models that represent queries and documents as vectors in a Euclidean space and rank by a notion of "distance," and *probabilistic* models that represent queries and documents as being drawn from probability distributions and rank by some probabilistic likelihood function. Of course, one could imagine countless other retrieval models, but we limit our discussion in this thesis to these three broad classes because historically they have been the most well explored. We discuss the first two in appendix B, and the third here.

Our work uses a *probabilistic modelling* approach to retrieval as its framework, a powerful base for further development of principled models. As the name implies, this approach involves building models (that is, probability distributions) representing queries and documents, defining some probability function of relevance, and using that function to perform ranking. There are basically two approaches that are commonly taken, the *classic* approach, and the *language modelling* approach.

We first describe the classic approach, looking specifically at one of the first probabilistic models, the *binary independence retrieval* (BIR) model. We then briefly look at language models. Chapter 3 will examine in-depth the a more sophisticated probabilistic model that serves as the baseline of our work.

Binary Independence Retrieval

The binary independence retrieval model is the first and arguably most influential probabilistic model of information retrieval, originally proposed by Robertson and Spärck Jones [17]. As originally conceived, the user would iteratively interact with the retrieval system, indicating whether the returned documents are relevant or irrelevant. The system would then incorporate that information into its ranking model to refine its model of relevance, and consequently the result set. This is the process of *relevance feedback*, which we return to later in this chapter. Until then, let us first examine how the initial set of documents is retrieved, before relevance information is gathered.

This model tries to estimate the probability that documents in the corpus are relevant to the user, conditioned on the user's input query. The model then assumes that the ideal answer set is one which maximizes the probability of relevance of the returned documents, which is done by ranking by the probability of relevance. Mathematically, if we let r denote a binary random variable for relevance and d a document, that means we rank by:

$$\frac{\Pr[r \mid d]}{\Pr[\neg r \mid d]}. \tag{2.1}$$

This is the *probability ranking principle* (PRP), which is further discussed in chapter 3. In the meantime, it suffices to say that this is equivalent to ranking by:

$$\frac{\Pr[d \mid r]}{\Pr[d \mid \neg r]}. \tag{2.2}$$

The next step is to break down the components of equation 2.2, that is, $\Pr[d \mid r]$ and $\Pr[d \mid \neg r]$. In the original BIR, we assume that documents are characterized by whether each index term occurs in them or not. We specifically disregard the frequency of occurrence of index terms. We also assume that index terms occur *independently* of each other—that is, whether a term w occurs in document d does not affect the probability of a different term w' occurring in d. Borrowing terminology from the previous section, let $\text{occurs}(w, d)$ be a binary function that is one if term w

occurs in document d and zero otherwise, D be the set of all documents (the corpus), and V be the set of all index terms (the vocabulary). Then we have:

$$\frac{\Pr[d \mid r]}{\Pr[d \mid \neg r]} = \frac{\prod_{w \in V \text{ and } \text{occurs}(w,d)=1} \Pr[w|r] \times \prod_{w \in V \text{ and } \text{occurs}(w,d)=0} (1 - \Pr[w|r])}{\prod_{w \in V \text{ and } \text{occurs}(w,d)=1} \Pr[w|\neg r] \times \prod_{w \in V \text{ and } \text{occurs}(w,d)=0} (1 - \Pr[w|\neg r])}. \tag{2.3}$$

Equation 2.3 simply decomposes the probability of a document into the probability of its index terms. It says that the probability of a document being (ir)relevant is the probability that each of its terms would appear in a (ir)relevant document, times the probability that every other term would not appear in a (ir)relevant document. By making the term independence assumption, we avoid any conditioning between the terms.

In practice, equation 2.3 involves the multiplication of a number of small probabilities, making the computation prone to underflow. To mitigate this effect, we generally use the logarithm of the ranking value (a transformation which maintains the ordering of documents), to turn the multiplications into additions:

$$\begin{aligned} \log \frac{\Pr[d \mid r]}{\Pr[d \mid \neg r]} = & \sum_{\substack{w \in V \text{ and} \\ \text{occurs}(w,d)=1}} (\log \Pr[w|r] - \log \Pr[w \mid \neg r]) \\ & + \sum_{\substack{w \in V \text{ and} \\ \text{occurs}(w,d)=0}} (\log(1 - \Pr[w|r]) - \log(1 - \Pr[w \mid \neg r])). \end{aligned} \tag{2.4}$$

The question remains of how we estimate $\Pr[w \mid r]$ and $\Pr[w \mid \neg r]$. Let us make the crude simplifying assumptions that initially, a relevant document has a 50-50 chance of containing each index term, and that the terms of irrelevant documents should look roughly like the background distribution of terms in the entire corpus. The first assumption is made because given no other information we can only guess a uniform probability of occurrence for each term. The second assumption is motivated by the fact that most documents are irrelevant to any particular query, so that the entire corpus is a good approximation to just the irrelevant documents. Mathematically, we

have:

$$\Pr[w \mid r] = 0.5 \tag{2.5}$$

$$\Pr[w \mid \neg r] = \frac{\sum_{d \in D} \text{occurs}(w, d)}{|D|}. \tag{2.6}$$

Again, equation 2.5 simply states that each index term is equally likely to show up or not in the relevant documents, and equation 2.6 states that a term should occur in the irrelevant documents with the same chance that it appears in a random corpus document.

The astute reader will notice at this point that we have completely disregarded the query from this retrieval process. The original BIR incorporated the query in a rather crude manner, by simply only applying the ranking to documents where the query terms show up, and having the user perform further relevance feedback to drill down on the actual relevant documents. A more sophisticated mechanism for incorporating the query would be to assume that it is generated by the relevant distribution, and using that assumption to create a better initial estimate for the relevant distribution. We explore this approach in the next chapter.

Language Models

A more recent probabilistic ranking scheme is the *language model*, which ranks documents according to $\Pr[q \mid d]$. That is, we train a model based on each individual document, giving us $|D|$ models (the eponymous language models). Then, we compute the probability that the query is generated from each of the models, and rank the documents by that likelihood.

Though not immediately obvious, the language modelling approach can also be formalized as maximizing the probability of relevance. For details of that derivation, refer to Lafferty and Zhai's work [18].

Language models have been shown to perform strong empirically, likely due to the increased amount of "training data" for each of the models (documents are generally much longer than queries and thus the models are much richer). However, their

theoretical validity has been questioned, and so for our theoretically-grounded work we have chosen to stay with a more classically-oriented probabilistic approach.

2.4 Query Reformulation

We can build arbitrarily sophisticated models for retrieval, but we are often limited by the richness of the query specification. If the user gives us a simplistic query, such as just one or two words, there is a limited amount of sophistication we build on top of that. To overcome this sparsity of data, we can try to *reformulate* the query after an initial pass at retrieval. One reformulation technique is to actually go back to the user and ask for additional information after the initial query. Because we do not want to expose users to the guts of a retrieval system, the request should be done in a "natural" way. That is, we must make the interactive request as straightforward and intuitive as possible. Alternatively, we could try to somehow use the initial retrieval results to make a better second pass of retrieval results.

2.4.1 Relevance Feedback

Relevance feedback is a popular reformulation technique that is both practical and relatively natural for users. In this approach, the user is presented with the initial set of retrieval results and is asked to explicitly judge whether each is relevant or irrelevant. We then use the information that the user provides to perform another retrieval pass, with a reformulated query that takes into account the relevance judgments. This information is taken into account by emphasizing the terms that appear in the judged-to-be-relevant documents, and deemphasizing terms in judged-to-be-irrelevant documents.

Relevance feedback can be repeated indefinitely, as long as the user is not satisfied with the set of results that he or she has gotten so far. Users find the relevance feedback cycle relatively natural, because it is simple to judge whether a document satisfies an information need or not. Intuitively, by "drilling down" on the documents that the user selects to be relevant, and moving away from those that are deemed

irrelevant, the system should be able to adapt its results toward the "correct" result set.

2.4.2 Pseudorelevance Feedback

Inspired by the idea behind relevance feedback, we can go a step farther and simply *assume* that the top documents returned by a first pass of the retrieval system are relevant. We then feed back these documents as positive examples and rerun the query with the expanded definition. In this approach, known as *pseudorelevance feedback*, we do not have to go through the intermediary step of actually asking the user for relevance judgments, so the actual interface presented to the user is still one iteration of query to result set. Hence, pseudorelevance feedback is a *local analysis* technique—it uses the initial result set as input into a second pass at retrieval. Local analysis is a reformulation technique that does not rely on any additional interface machinery. There are also *global analysis* techniques that rely on properties of the corpus as a whole, or even external data; we will not be discussing global analysis in this thesis.

Pseudorelevance feedback is a heuristic that has been shown to improve retrieval performance in most scenarios [4], and is appealing because we appear to get something for free—increased relevance without additional user workload.

2.4.3 Applied to the BIR Model

Generally, relevance feedback approaches can be applied to any model, but in different ways. To make our discussion concrete, in this section we discuss how we can *reweight* term probabilities in the BIR model when given feedback documents, either through explicit or pseudo relevance feedback. Relevance feedback is also popular for vector space models, where we use returned documents to *expand* the queries. See Efthimiadis's paper for further discussion of query expansion [9].

Recall the default term probability assignments for the BIR model, equations 2.5

and 2.6:

$$\Pr[w \mid r] = 0.5$$
$$\Pr[w \mid \neg r] = \frac{\sum_{d \in D} \text{occurs}(w, d)}{|D|}.$$

Let D_r be the set of documents that were selected to be relevant. Saying that a document is relevant means that other relevant documents should look like it—that is, the probability of a term occurring in a relevant document should be approximately equal to the probability that the already-selected relevant documents have that term. Similarly, we still assume that the corpus as a whole is representative of irrelevant documents, except for those documents that were marked relevant, so we remove those documents from our estimate for the irrelevant distribution. Mathematically, we have:

$$\Pr[w \mid r] = \frac{\sum_{d_r \in D_r} \text{occurs}(w, d_r)}{|D_r|} \tag{2.7}$$
$$\Pr[w \mid \neg r] = \frac{\sum_{d \in D} \text{occurs}(w, d) - \sum_{d_r \in D_r} \text{occurs}(w, d_r)}{|D| - |D_r|}. \tag{2.8}$$

We are not done yet, however. Assume that only relevant document d was selected—then the estimates for the relevant documents would assign the terms in d to have probability one, and terms that do not occur in d probability zero. This is obviously undesirable—we would not be able to retrieve any documents that did not exactly match document d's term occurrences! Hence, we'd like to *smooth* out our distribution estimate by reducing the very high probabilities and increasing the very low ones. A common smoothing technique is to add a small adjustment factor to all the probabilities to prevent any individual estimate from going to 0 or 1:

$$\Pr[w \mid r] = \frac{\sum_{d_r \in D_r} \text{occurs}(w, d_r) + \frac{\sum_{d \in D} \text{occurs}(w,d)}{|D|}}{|D_r| + 1} \tag{2.9}$$
$$\Pr[w \mid \neg r] = \frac{\sum_{d \in D} \text{occurs}(w, d) - \sum_{d_r \in D_r} \text{occurs}(w, d_r) + \frac{\sum_{d \in D} \text{occurs}(w,d)}{|D|}}{|D| - |D_r| + 1}. \tag{2.10}$$

Note that we disregarded the irrelevant documents that are fed back in this model. A more sophisticated model could explicitly use that information as well.

2.5 Summary

This chapter presented an overview of the retrieval pipeline, from the textual preprocessing to the index building and the retrieval model. Our focus was on the standard Binary Independent Relevance (BIR) probabilistic model, as a first taste of the kind of probabilistic machinery we will be using later. We also discussed relevance feedback, particularly in the context of BIR; relevance feedback will arise naturally out of some of our later work.

Chapter 3

Naïve Bayes Retrieval

In this chapter we focus specifically on a probabilistic Naïve Bayes model for retrieval, a simple model that we build on for our contributions. Our discussion will be like peeling an onion. We first begin from the outside, describing the scoring expression used to rank documents. This expression is defined in terms of various probabilities. We then proceed to look at how we compute these probabilities, by showing how to model them using multinomial distributions. Finally, we fill in the details of the distributions by presenting the estimation procedure that allows us to assign values to the distributions' parameters.

We view documents and queries as an unordered *bag of words*—that is, a document is completely characterized by the words it contains, and the frequency of those occurrences, without regard to the order those words occur in. Documents are ranked according to the probability of relevance conditioned on a query, which can be transformed via Bayes' Rule into an expression in terms likelihood of documents given that its relevant or irrelevant. To estimate these likelihoods, the model posits the existence of two *generative* probability distributions of documents, the *relevant* and *irrelevant* distributions. These distributions assign probabilities to bags of words (that is, new documents and queries). We assume that the distributions are from the *multinomial* family, with parameters estimated (learned) from the query term and the background corpus word distribution.

Throughout this discussion, we will be using some common notation. d will rep-

resent an arbitrary corpus document; q the query; r the relevance variable, as well as the generative relevant document model; and $\neg r$ the generative model for irrelevant documents. Other bits of terminology will be introduced as needed.

3.1 The Probability Ranking Principle

Let us begin by developing an intuitive understanding of ranking. We would like to develop a "black box" mechanism that finds the "best" ranked list of documents as a result set. One straightforward way to do this is to score each corpus document individually and independently, on how "good" that one document is for answering the query's underlying information need. Obviously, this assumption of independence is strong; the presence of one document in the results affects how useful the other documents are. However, it does simplify the entire ranking problem into one of scoring each corpus document. Once we have each document's score, we just return the documents in rank order, cutting off after a certain score threshold, or a fixed number of documents.

In the probabilistic IR context, the score of "goodness" is represented by a probability of relevance, and so ranking by score is equivalent to ranking by probability of relevance of each document:

$$\Pr[r \mid d, q]. \tag{3.1}$$

This is the canonical *probability ranking principle* (PRP) scheme. That is, the first result is the document with the highest relevance probability, the second result is the second highest, etc.

It is not immediately obviously how we calculate a value of equation 3.1 for an arbitrary document d. Let us use Bayes' Rule (see equation A.3) to reexpress the ranking equation in terms of probabilities of documents. Such a representation will be more convenient for modeling the underlying probability distributions later on.

Following the math, we have:

$$\begin{aligned}
\Pr[r \mid d, q] &= \frac{\Pr[d \mid r, q] \Pr[r \mid q]}{\Pr[d \mid r, q] \Pr[r \mid q] + \Pr[d \mid \neg r, q] \Pr[\neg r \mid q]} \\
&= \frac{1}{1 + \frac{\Pr[d|\neg r, q] \Pr[\neg r|q]}{\Pr[d|r,q] \Pr[r|q]}} && (3.2) \\
&\stackrel{\text{mt}}{=} \frac{\Pr[d \mid r, q] \Pr[r \mid q]}{\Pr[d \mid \neg r, q] \Pr[\neg r \mid q]} && (3.3) \\
&\stackrel{\text{mt}}{=} \frac{\Pr[d \mid r, q]}{\Pr[d \mid \neg r, q]} && (3.4) \\
&\stackrel{\text{mt}}{=} \log \Pr[d \mid r, q] - \log \Pr[d \mid \neg r, q]. && (3.5)
\end{aligned}$$

The steps of equality indicated by $\stackrel{\text{mt}}{=}$ are *monotonic transformations*—because we are only using the values of the ranking formula as a comparison value between documents, we can transform the values in any way that keeps the ordering the same. More precisely, we can replace the expression with any increasing function of that same expression. From equation 3.2 to 3.3, we do an algebraic transform. From 3.3 to 3.4, we remove $\frac{\Pr[r|q]}{\Pr[\neg r|q]}$, which is constant independent of the specific document. Finally, from 3.4 to 3.5 we take the logarithm of the ranking equation, which will simplify our later ranking by turning multiplications into additions. The final expression, equation 3.5, is called a *ranking value.*

We have now expressed the original objective, equation 3.1, in terms of two probabilities, conditioned on relevance or irrelevance. At first glance this does not seem to simplify our problem—we have transformed a problem of trying to find one probability into one of finding two. We now turn to how these probabilities are computed, by examining the underlying assumptions behind how documents are generated.

3.2 Generative Multinomial Models

We can express the probabilities of a document conditioned on relevance or irrelevance *generatively.* In this section, we describe in general what a generative model is, and specifically what a multinomial model, the model family that we will be using, looks

like. The next section then applies this general modelling knowledge to the case of our particular relevance and irrelevance models.

A generative model assigns probabilities to arbitrary documents by positing that they are sampled from an underlying distribution. This distribution in some sense is an approximation of the writing process for documents. We now would like to recover a "guess" as to what the underlying distribution looks like, so that we could then directly compute probabilities of documents being generated from that distribution. This guess we make on the basis of the query, information from the corpus, and possibly other external data sources.

"Guessing" the generative model is a two-step process. First, we need to decide the *family* of models. Roughly, this means we must determine the basic shape of the distribution that we *assume* the terms should follow. Second, we must choose a specific model from this family by setting the *parameters* of the model. These parameters are generally numerical values that tune the precise formulae of the distribution. Very approximately, the family defines the qualitative features of the model, while the parameters then tune the precise quantitative form.

Let us first examine the family of models. In this thesis, we will be limiting our family to a particularly simple one, the *multinomial distribution.* The generation process for a draw from a multinomial can be viewed as a series of rolls of a n-sided die, where n is the number of different possible outcomes. For example, to generate a document of length l, we could roll a $|V|$-sided die (V being the vocabulary), where each face is one of the index terms, l times. The parameters of the multinomial are the probabilities of each of the die faces. The multinomial distribution is considered *multivariate* because it represents the probability distribution for a set of random variables jointly. Hence, *one* sample from an n-dimensional multinomial distribution provides values for *all* n random variables.

Mathematically, the probability density function for the multinomial distribution of dimension n over random variables X_1 through X_n is

$$\mathrm{Mu}_n[X_1 = x_1, \ldots, X_n = x_n; \theta_1, \ldots, \theta_n] = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} \theta_i^{x_i}. \tag{3.6}$$

X_i specifies the number of times die face i shows up, and θ_i is the probability of face i. Specifically in the case of document generation, each X_i is the frequency of index term i, and $\sum_{i=1}^{n} x_i$ is the total term count, or document length. Because θ_i represents the probability of each face, we must also constrain their sum to be one, that is, $\sum_{i=1}^{n} \theta_i = 1$.

Returning to the probability ranking principle's ranking value, recall that we assume the existence of generative models for both relevant and irrelevant documents. Let θ^{rel} be the parameter vector of the relevant distribution and θ^{irr} be the parameter vector for the irrelevant distribution. Let document d have term frequencies w_1, w_2, ..., w_n for index terms 1 through n. Now we can plug equation 3.6 into equation 3.5 to obtain a formulation for the ranking value in terms of the multinomial distribution parameters:

$$\begin{aligned}
\log \Pr[d \mid r, q] - \log \Pr[d \mid \neg r, q] &= \log \mathrm{Mu}_n[w_1, \ldots, w_n; \theta^{\text{rel}}] - \log \mathrm{Mu}_n[w_1, \ldots, w_n; \theta^{\text{irr}}] \\
&= \log \frac{(\sum_{i=1}^{n} w_i)!}{\prod_{i=1}^{n} w_i!} \prod_{i=1}^{n} (\theta_i^{\text{rel}})^{w_i} - \log \frac{(\sum_{i=1}^{n} w_i)!}{\prod_{i=1}^{n} w_i!} \prod_{i=1}^{n} (\theta_i^{\text{irr}})^{w_i} \\
&= \sum_{i=1}^{n} w_i \left(\log \theta_i^{\text{rel}} - \log \theta_i^{\text{irr}} \right) \qquad (3.7)
\end{aligned}$$

We have dropped the query term q in this derivation, but as we will see later, q helps us determine how to set the parameter vectors θ.

Given a selection of model family, we now turn to the second problem. We require a procedure to set the values of parameters, or an *estimation* procedure. For the multinomial distribution, this means we need a way to set θ^{rel} and θ^{irr} for each index term. We first look at general approaches to estimation (independent of model family), and how we train from *examples*, data which we assume was drawn from the unknown distribution. We use this data, sometimes along with background knowledge, to infer the settings of the parameters. We then apply these estimation principles to the multinomial distribution specifically, so we can perform estimation in the context of document ranking.

3.3 Estimation

In general, in the estimation process we try to infer "reasonable" values of distribution parameters based on training data or prior knowledge. Training data refers to samples which we assume are drawn from the distribution. Prior knowledge is generally some belief that we hold about the parameters before we see any data. There are a variety of mechanisms to do estimation, some of which are specific to the distribution family, and some of which are generally applicable. In this section we describe two of the most commonly used statistical estimators.

Notation-wise, parameter estimates are usually indicated by a hat over the parameter, e.g., $\hat{\theta}$.

3.3.1 Maximum Likelihood

Maximum likelihood (ML) estimation sets the parameters of a distribution by maximizing the probability of the *joint likelihood* of the given training examples. Simply stated, different choices of parameters θ will lead to different probabilities that the training examples S were generated from the model. Our θ estimate is then the choice that produces the highest such probability. Mathematically, we would like to maximize a *likelihood function* based on the m training examples S_1, S_2, ..., S_m, which is the probability of the data, as a function of the model parameters:

$$\hat{\theta}_{\mathrm{ML}} = \operatorname*{argmax}_{\theta} L(\theta) \tag{3.8}$$

$$L(\theta) = \Pr[S_1, \ldots, S_m; \theta]. \tag{3.9}$$

If we make the common assumption that each training example S_i was *independently* sampled from the underlying distribution, we can then turn equation 3.9 into a product:

$$L(\theta) = \prod_{i=1}^{m} \Pr[S_i; \theta]. \tag{3.10}$$

As is often the case with multiplicative probabilities, we can simplify the computation by taking logarithms, which is a monotonic transformation, to obtain a *log likelihood* function:

$$\ell(\theta) = \sum_{i=1}^{m} \log \Pr[S_i; \theta]. \tag{3.11}$$

As the name *maximum* likelihood implies, we proceed to find the global optimum of the (log) likelihood function. Depending on the specific algebraic form of equation 3.11, this may either be possible analytically (by finding the zeros of the partial derivatives with respect to each parameter), or may be done with numerical methods.

In the case of the multinomial distribution, a straightforward derivation (see appendix C) shows that the appropriate maximum likelihood estimate for parameter $1 \leq k \leq n$ is:

$$\hat{\theta}_k = \frac{\sum_{j=1}^{m} x_k^j}{\sum_{i=1}^{n} \sum_{j=1}^{m} x_i^j}. \tag{3.12}$$

Equation 3.12 is an appealing result—it tells us that the parameters, which we originally motivated as being the probability of a particular outcome, should be set according to how frequently we observe that outcome in the empirical training data.

Maximum likelihood estimation is very popular and used in many disciplines. ML estimation is particularly robust when training data is abundant. However, consider the case when we may not have many training examples—indeed, if we have one, or even zero, training examples. Maximum likelihood would not be able to do anything useful in the latter case at all, and in the former, is likely to make very bad estimates. To see why, consider trying to train a model for a fair six-sided die based on one roll—no matter what that roll is, we would build a model that would assign full probability to one face, and zero to the others. This is clearly an undesirable estimate, particularly if we have some pre-existing belief about the model in question (for example, that we believe the die to be fair). The next estimation technique we examine provides a principled, quasi-Bayesian approach to combining prior belief and empirical observations.

3.3.2 Maximum a Posteriori

Maximum a Posteriori (MAP) estimation provides us more flexibility than maximum likelihood estimation, by allowing us to specify a *prior* distribution over the parameters. We begin by directly trying to optimize the probability of the parameters given the training data—essentially, we are treating the parameters as a random variable in themselves. The distribution of the parameters given the data is called the *posterior* distribution—hence the name of the technique, which is to use the mode of the posterior distribution as the point estimate for the parameters. An application of Bayes' Rule yields an expression that incorporates the probability of the data given the parameters, and the unconditional (prior) probability of the parameters. The estimation technique then tells us to choose the setting of the parameters that maximizes this expression.

As before, let the m training examples be S_1, S_2, ..., S_m. The MAP estimator tells us estimate the parameters according to the mode of the posterior distribution, that is,

$$\hat{\theta}_{\text{MAP}} = \operatorname*{argmax}_{\theta} \Pr[\theta \mid S_1, \ldots, S_m]. \tag{3.13}$$

An application of Bayes' Rule to the posterior probability yields:

$$\Pr[\theta \mid S_1, \ldots, S_m] = \frac{\Pr[S_1, \ldots, S_m \mid \theta] \Pr[\theta]}{\int \Pr[S_1, \ldots, S_m \mid \theta'] \Pr[\theta'] d\theta'} \tag{3.14}$$

$$\stackrel{\text{mt}}{=} \Pr[S_1, \ldots, S_m \mid \theta] \Pr[\theta] \tag{3.15}$$

$$\stackrel{\text{mt}}{=} \log \Pr[S_1, \ldots, S_m \mid \theta] + \log \Pr[\theta]. \tag{3.16}$$

We are able to make the simplification from equation 3.14 to equation 3.15 because the denominator is a normalization factor that is independent of the specific choice of parameters. The final step is taking a logarithm. Contrast equation 3.15 with equation 3.9—we are maximizing the same likelihood function as in maximum likelihood, except with an additional term, $\Pr[\theta]$, the prior probability of the parameters. If we assume independence between training examples, we can further simplify equa-

tion 3.16 into:

$$\Pr[\theta \mid S_1, \ldots, S_m] = \sum_{i=1}^{m} \log \Pr[S_i \mid \theta] + \log \Pr[\theta]. \tag{3.17}$$

We can handle the $\Pr[S_i \mid \theta]$ in the same way as with maximum likelihood—but we have a new term for the prior, $\Pr[\theta]$.

How do we model the prior distribution? The prior allows us to specify some preexisting belief about the values of the parameters. Returning to the multinomial six-sided die example, we could specify a prior that the die is fair—that is, $\theta_1 = \theta_2 = \ldots = \theta_6 = 1/6$. However, it is not enough to specify point values for the prior estimates—we need to specify an entire distribution over the parameters θ. For example, if we strongly believe that the die is close to fair, we could specify a distribution that is tightly centered around $1/6$, which assigns little probability to θ estimates close to zero or one. By using this prior, the MAP estimator would "penalize" estimates of θ far away from $1/6$, even if such a model is likely to generate the training examples. If we had specified a uniform distribution over all the possible values of θ (in the die example, a uniform over the interval $[0, 1]$ in six dimensions), then we call the prior *uniform*, and the MAP estimate ends up being the same as the maximum likelihood estimate.

The particular form of the prior could be arbitrarily sophisticated, and in the general case the posterior probability could then also be very complex. However, many distributions have a so-called *conjugate prior*—a family of distributions that takes the same algebraic form as the likelihood of the data. More precisely, a distribution family D is a conjugate prior for a distribution family D' if the posterior distribution of the parameters of D' given the data is also from the family D. In the case of an n-dimensional multinomial distribution, the conjugate prior is an n-dimensional *Dirichlet distribution*, a continuous multivariate distribution over the interval $[0, 1]$ in

each dimension, constrained so that the n random variables have to sum to one:

$$\mathrm{Dir}_n[X_1 = x_1, \ldots, X_n = x_n; \alpha_1, \ldots, \alpha_n] = \begin{cases} \frac{\Gamma(\sum_{i=1}^n \alpha_i)}{\prod_{i=1}^n \Gamma(\alpha_i)} \prod_{i=1}^n x_i^{\alpha_i - 1} & \text{if } \sum_{i=1}^n x_i = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

In this expression, $\Gamma(x)$ is the *Gamma function*, a generalization of the factorial to real (and complex) numbers. The constant in front of the product is just a normalization term, and the constraint ensures that the probability function is zero unless the data sums to one. The Dirichlet distribution is itself parameterized on values α—to distinguish them from the regular parameters θ, we will refer to the α vector as the *hyperparameters.*

We now have the additional responsibility of assigning values to the hyperparameters. We should set them directly according to our prior belief. In the particular case of the Dirichlet distribution, it turns out that the expected value of the ith random variable is equal to $\alpha_i / \sum_{j=1}^n \alpha_j$. Hence, we set each α_i according to what we expect θ_i to be, irrespective of training data. The scale of the sum of the αs determines the *strength* of the prior—which we will examine more closely shortly.

Referring to the derivation in appendix C, the maximum a posteriori estimate for parameter $1 \leq k \leq n$ is:

$$\hat{\theta}_k = \frac{\sum_{j=1}^m x_k^j + \alpha_i - 1}{\sum_{i=1}^n \sum_{j=1}^m x_i^j + \sum_{i=1}^n \alpha_i - n}. \tag{3.19}$$

Using the Dirichlet distribution as the prior for the multinomial distribution has a particularly appealing intuitive rationale. Note that the difference between the maximum likelihood and maximum a posteriori estimates for θ is the additional α_i hyperparameter term. Thus, the α_i can be interpreted as *prior examples* for each dimension of the multinomial distribution. By setting the α_is, we are specifying a belief via "fake" training examples. We can scale the α_is together higher or lower to indicate how strongly we believe in our prior compared to training examples. As we get more actual training examples, the α_i terms shrink compared to the $\sum_{j=1}^m x_k^j$

terms; in the limit, the MAP estimate asymptotically approaches the ML estimate as we get more training examples, no matter how we had set our prior.

3.3.3 Other Estimation Techniques

Maximum likelihood and maximum a posteriori estimators are both known as *point estimators*, because they compute one value (guess) of the parameter settings that is then used as the definitive estimate. A fully Bayesian technique, in contrast, would model the uncertainty in the estimates by representing them as probability distributions. These estimation techniques tend to be much more computationally and conceptually difficult, but provide an interesting route of future work.

3.4 Estimating Relevance and Irrelevance Models

Now let us take our knowledge of estimation and apply it to retrieval. We will be taking the MAP approach. Recall that as originally formulated, we wanted to build models to estimate the terms $\Pr[d \mid r, q]$ and $\Pr[d \mid \neg r, q]$, so that we need to estimate two distributions, the relevant document distribution and the irrelevant document distribution. So far, we have represented both distributions as multinomials. We will make the following additional assumptions, which will then guide us to the appropriate training routine for these models:

1. The query is an example of a relevant document.

2. Without any other information, both the relevant and irrelevant term distributions should appear similar to the overall corpus term distribution.

3. Knowledge of the query does not tell us anything about the appearance of the irrelevant documents.

The first assumption implies that q can be interpreted as a *training example* from the distribution for r. The second assumption tells us to set the prior distributions of both r and $\neg r$ according to the overall term distribution of the corpus. The

third assumption indicates that we can drop the conditioning on the query from the expression for the irrelevant distribution—that is, $\Pr[d \mid \neg r, q] = \Pr[d \mid \neg r]$.

The relevant distribution is thus trained with a MAP estimator, using a Dirichlet distribution centered on the corpus word distribution, and the query as the one training example. The irrelevant distribution does not have any training examples, and hence the best we can do is set it to the prior, the background word distribution.

The final question we need to address is how updates to the models are weighted against the model prior distributions. Mathematically, we know how to set the α_i parameters of the Dirichlet prior with respect to each other, but we do not know to scale the sum of the parameters as a whole. Generally, we will tune this weight (the sum of the α_is) as part of the experimental setup. To maintain the robustness of the experiments, we use either a held out development set or cross-validation. In the former, we reserve a portion of the experimental data (for example, some percentage of the test queries) for use

The reader may be somewhat disappointed at this point at our extensive development of the background statistical machinery, only to use it briefly with one training example. Fear not, intrepid reader, for we will be using these techniques much more extensively when we talk about our more sophisticated objective functions later in this thesis.

3.5 Accounting for Model Deficiencies

The model we have proposed is obviously makes many simplifying assumptions about the way documents are generated. The focus of this thesis is not on the models *per se*, but rather the way the model is used to perform ranking. Therefore, we have chosen a particularly simple underlying model that is easy to understand and performs sufficiently well empirically.

However, we do bow to using some heuristics to improve the empirical performance of the baseline model. Taking a note from previous work [21], we have chosen to perform a *log transformation* of the term frequencies—that is, we replace term oc-

currences x_i with $\log x_i + 1$. This modification ensures that documents with the same word repeated over and over are not unduly penalized, because in fact documents about one subject generally *do* have multiple occurrences of the same word. The multinomial model is indeed notorious for dealing poorly with burstiness of words. This heuristic does not affect the theoretical reasoning behind our model, though technically, the multinomial distribution applies only to integer random variables, and using the logarithm will obviously produce nonintegral results. However, by using the MAP estimator, the Dirichlet posterior is still unchanged, and can be used as described. We believe this slight theoretical muddling is worthwhile in the face of significant empirical improvement.

3.6 Summary

This chapter has been a firehose of theoretical knowledge, so let us summarize our discussion on ranking.

The probability ranking principle states that we rank by the probability of relevance of a document, given a query. An application of Bayes' Rule transforms this ranking into the ratio between the probability of the document being generated from the relevant document distribution, to the probability of it being generated from the irrelevant document distribution. Both of these distributions we assume to multinomial. We use a maximum a posteriori approach to estimate the parameters of these distributions. The relevant distribution is trained with a prior centered around the background corpus distribution, with the query taken as the one training example. The irrelevant distribution is informed only by its prior, also the background corpus distribution. Finally, we indicated that in reality we will be using a transformed version of the term frequencies, to make up for a deficiency of the multinomial model.

Chapter 4

Evaluation

For IR to be a science, practitioners must develop and use rigorous, consistent, standardized *evaluation* techniques. Now that we have discussed the actual mechanics of retrieval, we turn our attention to the evaluation problem. Note that evaluation in this context is taken to meant the "quality" of the results returned by the search system, not the time and space efficiency of the system, though obviously efficiency is an important practical consideration.

Evaluation is particularly important in our work, because we draw a fundamental connection between the mechanism used for evaluation, the *metrics*, and the algorithm for retrieval. A metric is a numerical scoring functions that judges the quality of a retrieval system's results, usually against a gold standard that is human defined. Different metrics measure different notions of result set quality. For example, some metrics such as the canonical precision and recall encourage result sets with many relevant results, whereas others such as search length and reciprocal rank are concerned with where the first relevant result is in a result set. In particular, we will emphasize a new class of metrics that embodies a new perspective on result set relevance that represents a broad possibility of user needs and is easily representable in our retrieval framework.

4.1 Data

Before embarking on our exploration of metrics, we first discuss standard evaluation *data* that is commonly used in IR, focusing on the TREC dataset that we use for our experiments. For evaluations of two separate systems to be comparable, they must be tested on sample corpora and queries that are "similar." For example, if we tested system A on a much more difficult set of queries than system B, and find that system B performs better, if would be unfair to claim that system B is better. Clearly, the way we can control the independent variable of the dataset is to use the *same* test corpora and queries for each system under consideration. This simple realization in IR has led to the development of widely distributed standard corpora, associated test queries, and their respective relevance judgments.

The most widely used of these standard evaluation datasets is data published annually at the Text REtrieval Conference (TREC) [14, 25]. TREC publishes a combination of new corpora and new queries (with relevance judgments) every year. This data is used for competition between IR systems at the conference itself, and as a golden standard for evaluation of retrieval systems in general. TREC runs multiple *tracks* of competition, which emphasize different retrieval tasks (such as question/answering, web search, or retrieval robustness).

We will be using data from various TREC tracks to perform evaluation of our techniques. TREC uses its own unique nomenclature. Queries are known as *topics* and are uniquely numbered. Most topics come with a title, description, and narrative, each of which more precisely describes the information request than the last. Titles are generally short, two or three word phrases; descriptions may be one sentence or longer phrase; and narratives are paragraph-length instructions on what precisely the information need is. By using different combinations of the topic components as input to the retrieval system, we can vary the exact query input to the retrieval system.

Each topic will also have associated with it a set of relevance judgments known as *qrels*.[1] Qrels are simply mappings between topics and documents—generally a tuple

[1]Some topics have multiple qrel sets, for different retrieval tracks.

specifying the topic number, the document number, and a human-made judgment on the relevance of that document to that topic. The judgments are usually binary—that is, either the document is relevant to that topic, or it is not—but for some tasks is ternary, indicating irrelevance, some relevance, or high relevance.

Now that we have standard sets of evaluation data, we turn our attention to how we use the qrels to score our retrieval systems' results, by looking at metrics.

4.2 Metrics

A metric is a function that assigns a score to a retrieval system, given a set of queries, a corpus, and relevance judgments between the queries and corpus documents and/or document sets. In general, given one set of test queries, a metric imposes a preference ordering between retrieval systems. That is, the retrieval system that scores highest (alternatively, lowest) on a metric is the "preferred" system. Different metrics capture different notions of result set quality, and often a system that is optimal for one metric may not be for another.

Most metrics are computed for each individual query, and its corresponding relevance judgments. Metrics are generally averaged between individual queries by taking the arithmetic mean. However, there are some metrics which explicitly specify that the averaging is done differently, such as the *geometric mean average precision* metric. We will not be discussing those metrics in this thesis. For the purposes of our discussion, each of the metrics below should be averaged via arithmetic mean for the overall score of a system.

We will examine some of the commonly used metrics in the IR literature. These metrics assume that we have a set of relevance judgments on the test queries, such as those used for the TREC experiments. All the metrics we will be discussing (save one) only utilize binary relevance judgments. Throughout this section, we will use R to indicate the set of relevant documents as judged by annotators, and $d_1, \ldots, d_n$ to indicate the documents of a result set of size n in rank order. As a shorthand, D_i will indicate indicate the set of documents in the top i. The indicator function $I_R(d)$

evaluates to one if d is in set R, and zero otherwise.

4.2.1 Precision and Recall, and Variants Thereof

Precision at n is defined as the fraction of the top n retrieved documents that are in the relevant set:

$$P_n(D_n, R) = \frac{\sum_{i=1}^{n} I_R(d_i)}{n}. \tag{4.1}$$

Recall at n is defined as the fraction of all relevant documents that were retrieved as the top n results:

$$R_n(D_n, R) = \frac{\sum_{i=1}^{n} I_R(d_i)}{|R|}. \tag{4.2}$$

As previously discussed, for evaluation over multiple queries, we take the arithmetic mean of the precision or recall values.

Precision and recall are two of the most canonical metrics in IR, and indeed present an interesting tradeoff. A system that tries to cast a wide net for results will tend to get a lot of relevant results, but also a lot of irrelevant ones (false positives)—high recall and low precision. Conversely, a system that tries to focus narrowly on just finding "right" results will tend to have a lot of false negatives—high precision and low recall. To capture this tradeoff, precision and recall are often used in conjunction via a *precision versus recall* graph. To construct such a graph, we find the rank level at which the result set reaches x recall, generally using the values $x = 0.0, 0.1, \ldots, 0.9, 1.0$. We then compute the precision at each of those rank levels, interpolating as necessary, and graph those values against the recall thresholds. Note that recall is an increasing function of n, whereas precision is decreasing, so such a graph should be nonincreasing as a function of recall. In this thesis, we will focus on optimization of single-valued metrics, and thus focus on the precision and recall metrics defined at rank cutoffs, rather than the precision-recall graphs.

These two metrics have spawned many variations that have attempted to better capture the notion of result set quality. *R-precision* is precision at the rank level of the number of relevant documents (assuming that we retrieve at least that many

results), that is:

$$RP(D_n, R) = P_{|R|}(D_{|R|}, R) = \frac{\sum_{i=1}^{|R|} I_R(d_i)}{|R|}. \tag{4.3}$$

In some sense, R-precision helps us choose what rank level cutoff to evaluate at, rather than arbitrarily pegging it at some n.

Another variant of precision is *mean average precision* (MAP), which is widely used and reported in the retrieval literature, such as in TREC. Indeed, it is often considered to be *the* standard metric for measuring retrieval performance, though recent work has called MAP's validity into question [24]. The average precision for a single query is the arithmetic mean of the precision values at the rank of each relevant result, up to a rank cutoff n (that can be the size of the entire corpus). Mathematically,

$$AP_n(D_n, R) = \frac{\sum_{i=1}^{n} I_R(d_i) P_i(D_i, R)}{|R|} = \frac{\sum_{i=1}^{n} I_R(d_i) \sum_{j=1}^{i} I_R(d_j)/i}{|R|}. \tag{4.4}$$

MAP rewards systems that return more relevant results early on, because the precision at those ranks will be higher.

There are other single value metrics that are closely related to precision and recall; generally these metrics try to combine and trade off precision and recall in one value. The *F-measure at n* is one such combination—it is simply the harmonic mean of precision and recall at n:

$$F_n(D_n, R) = \frac{2}{1/P_n(D_n, R) + 1/R_n(D_n, R)} = \frac{2\sum_{i=1}^{n} I_R(d_i)}{n + |R|}. \tag{4.5}$$

We can further generalize the F-measure by specifying that we'd like a tunable parameter that allows us to weight the relative importance of precision versus recall. The *E-measure at n* does exactly that, by adding a degree of freedom to the metric, b, that explicitly trades off precision and recall:

$$E_{n,b}(D_n, R) = 1 - \frac{b^2 + 1}{b^2/P_n(D_n, R) + 1/R_n(D_n, R)} = 1 - \frac{(b^2 + 1)\sum_{i=1}^{n} I_R(d_i)}{b^2 n + |R|}. \tag{4.6}$$

When $b = 1$, the E-measure reduces to one minus the F-measure. For $b > 1$, we weight precision higher than recall, and conversely for $b < 1$.

4.2.2 Reciprocal Rank and Search Length

We now turn our attention away from metrics related to precision and recall, towards metrics that focus more specifically on the location of the first relevant result. These metrics are used frequently for tasks where one relevant result is likely to answer the user's information request. For example, in web IR recall is difficult to compute, and likely to be meaningless—it does not matter that we return all web pages on a particular topic for a given query. Instead, it matters that we return *some* relevant result as soon as possible.

First, *reciprocal rank* (RR) is the reciprocal of the rank position of the *first* relevant result of the result set. We can define a cutoff point for reciprocal rank at n, where if there are no relevant results in the top n, the reciprocal rank is zero. The precise mathematical formulation for RR is:

$$RR_n(D_n, R) = I_R(d_1) + \sum_{i=2}^{n} \frac{I_R(d_i)}{i} \prod_{j=1}^{i-1} (1 - I_R(d_j)). \tag{4.7}$$

The average of reciprocal rank over multiple queries is typically referred to as mean reciprocal rank (MRR).

A related metric that also captures the notion of finding a good result quickly is *search length*. As originally defined, the concept of search length represented the irrelevant number of documents that a user would need to look through before having their information need completely satisfied. In the simplest case, if we assume that the user only needs one relevant document to satisfy his or her information need, then the search length is simply the rank of the first relevant result, minus one. If we have a result set of size n that is completely irrelevant, we take the search length to be n.

$$SL_n(D_n, R) = \sum_{i=2}^{n} (i-1) I_R(d_i) \prod_{j=1}^{i-1} (1 - I_R(d_j)) + n \prod_{i=1}^{n} (1 - I_R(d_i)). \tag{4.8}$$

Search length is closely related to reciprocal rank for evaluation of an individual query, but over multiple queries the two metrics average out differently. Specifically, the incremental cost of the first relevant document dropping a position in the ranking diminishes as a function of the ranking for reciprocal rank, but not search length. Reciprocal rank is a better metric to average over, because we do not want one very bad result set (where the first relevant result has a very high rank) to skew the overall average. However, search length better represents the overhead a user incurs in wading through irrelevant results for one query. Interestingly, if we took a harmonic mean over search lengths, the result would be effectively the reciprocal of the arithmetic mean of reciprocal ranks.

4.2.3 Instance Recall

Let us take a brief break from the notion of binary relevance for our next metric. One recently explored scenario, especially in the context of interactive retrieval systems, is *instance retrieval.* In this setting, each query is judged to have multiple *instances*, also known as aspects or subtopics, associated with it. These different instances correspond to different interpretations of the query. Each document and query pair is annotated not just with one binary relevance judgment, but instead with a binary judgment for each instance. Note that each query has its own unique set of instances, distinct from the other queries, and that the number of instances is variable. The number of subtopics for any given query is unknown to the retrieval system. To be more formal, each query has k subtopics, there are k separate, possibly overlapping, relevant sets R_1 through R_k, each of which is the set of documents that are relevant for that particular instance. As an example, consider the query "Trojan" on a web search engine. Trojan can refer to a citizen of Troy (made famous by the Trojan war), a computer Trojan horse, a brand of condoms, a former automobile manufacturer, a kind of asteroid or moon, or a music record label. When further disambiguating knowledge is unavailable, we could imagine that any of these interpretations is plausible. The corpus documents may address one or more of these interpretations. *Instance recall* at n measures how many unique such instances are accounted for in

the top n results:

$$IR_{n,k}(D_n, R_1, \ldots, R_k) = \frac{\sum_{j=1}^{k} \left(1 - \prod_{i=1}^{n}(1 - I_{R_j}(d_i))\right)}{k}. \tag{4.9}$$

A system that has high instance recall will retrieve many different interpretations of a query. Thus, instance recall is a natural proxy for the notion of diversity, a point we will demonstrate in our later experiments.

4.2.4 Other Metrics

The metrics that we have discussed are only one way of evaluating retrieval systems, and rely on batch processing of queries with binary of otherwise discrete relevant judgments. Other metrics proposed in the literature go beyond these limitations—for example, *coverage* and *novelty* rely on knowledge of what the user already knows or does not know, *relative recall* and *recall effort* depend on how many relevant results the user expected, *S-precision* and *S-recall* are more sophisticated ways of measuring relevance in the instance retrieval case [29]. Furthermore, there is an entire class of evaluation which emphasizes user studies and overall user satisfaction, rather than using numerical measures as a proxy. These metrics include, for example, *task completion time*, the amount of time a user takes to finish a defined task that involves retrieval, and *user satisfaction*, the self-reported experience of a user interacting with the retrieval system. We hope to consider some of these other metrics in future work.

4.3 A New Metric: k-call at Rank n

Now we return to the notion of binary relevance once again, and introduce our own new class of metrics. *k-call at rank n* for a single query, is one if at least k of the top n results are relevant, and zero otherwise. Mathematically, this states:

$$C_{n,k}(D_n, R) = \left\lceil \sum_{i=1}^{n} I_R(d_i) \geq k \right\rceil. \tag{4.10}$$

$[\cdot]$ is the Iverson bracket—for a logical proposition P, $[P] = 1$ if P is true and 0 otherwise. We can rewrite this metric without the Iverson bracket, but the expression is much more complex, except in the cases of $k = 1$ and $k = n$:

$$C_{n,1}(D_n, R) = 1 - \prod_{i=1}^{n}(1 - I_R(d_i)), \tag{4.11}$$

$$C_{n,n}(D_n, R) = \prod_{i=1}^{n} I_R(d_i). \tag{4.12}$$

A more intuitive equivalent formulation of the 1-call and n-call at n metrics is in terms of logical statements. If we take $I_R(d_i)$ to be a propositional statement indicating the relevance of the i document in the ranking, then we can define the metrics as logical true/false statements. 1-call at n requires that at least one document of the top n is relevant; either the first document is relevant, *or* the second document, *or* the third, etc. Similarly, n-call at n requires that the first document is relevant, *and* the second, *and* the third, etc.

$$C_{n,1}(D_n, R) = \bigcup_{i=1}^{n} I_R(d_i), \tag{4.13}$$

$$C_{n,n}(D_n, R) = \bigcap_{i=1}^{n} I_R(d_i). \tag{4.14}$$

Why this new metric? There are a few advantages of using this metric, particularly in our new retrieval framework. First, the extreme cases, of 1-call at n and n-call at n, capture particularly well the objectives of wanting one relevant result and wanting all relevant results, respectively. The former is a natural metric for use with a retrieval system that, for example, presents pages of results at a time, such as a web search engine. We would like a relevant result to appear on the first page, and 1-call directly measures for that. The latter is a notion of "perfect precision"—when it is critical that our retrieval system makes no errors, we reward only for returning a result set that is *entirely* relevant. Values of k between 1 and n allow us to do a risk/reward tradeoff. Lower k means we would be happier with fewer relevant results, and thus we can take more risk in choosing documents for the result set, whereas higher k implies

the opposite. Furthermore, the k-call metric is binary for an individual query, which makes it particularly amenable for use in our retrieval framework, as we will see later on.

4.4 Summary

In this chapter we have discussed two facets of evaluation, data and metrics. Standardized annotated datasets are important for retrieval evaluation and will be used in this work. The actual evaluation mechanism is represented numerically by metrics, which are quantitative measures of retrieval performance. We introduced a wide array of metrics, and introduced a new metric k-call that is important for the remainder of our work.

Chapter 5

The Expected Metric Principle

The last two chapters have developed the tools that we will now use to describe our novel contributions to probabilistic retrieval. In this chapter, we generalize the previously defined Probability Ranking Principle (PRP) to the *Expected Metric Principle* (EMP). In a nutshell, the EMP asserts that given a probabilistic model of relevance and a single-valued numeric metric, a ranking should attempt to maximize (or minimize) the *expected value* of the metric over possible rankings of results.

Formally, suppose we are given a user information request q, a corpus C, and a metric $\mathcal{M}(D, \mathcal{R})$ that scores result sets D according to relevance judgment information $\mathcal{R}$. Assume that higher values of this metric indicate a better retrieval system (a "higher-is-better" metric as opposed to a "lower-is-better" one). Then we select the "best" ranked result list D_n of n documents according to:

$$D_n = \operatorname*{argmax}_D E[\mathcal{M}(D, \mathcal{R}) \mid q]. \tag{5.1}$$

In particular, we refer to the argument being optimized, $E[\mathcal{M}(D, \mathcal{R}) \mid q]$, as the *objective function.* If the metric is a "lower-is-better", we instead find the minimum of the objective function. The EMP therefore reduces the problem of ranking to a problem of optimization—finding the document list of size n that produces the optimal value for the objective function.

There are a few points that we need to address. First, over exactly what search

space are we performing this discrete optimization? We explicitly specified the size of our result set, n, in our EMP formulation. Obviously, we do not want to duplicate documents in a result set either. Our space is therefore all unique orderings of n documents drawn without replacement from the corpus C, of which there are $P(|C|, n)$ possibilities, where $P(x, y) = x!/(x-y)!$ is the permutation function. In the extreme case, if we wanted to rank the entire corpus (as we can easily do with PRP), we would need to consider $|C|!$ unique result sets.

Second, how should n be selected? This question is as much connected to the interface as it is to the retrieval algorithm—a web search engine, for example, may have room for ten results, in which case we should optimize for metrics that are computed on the basis of the top ten. For metrics which can be computed over arbitrarily sized result sets, we could consider allowing n to vary as well, and maximizing over all result sets of different sizes. Such an approach is more computationally expensive, but may be beneficial or even necessary for the optimization of more sophisticated metrics than the ones we consider in this thesis. Finally, many metrics could require potentially a ranking of the entire result set, such as a pathological case of MRR where there is only one relevant document and it comes last in the ranking; in this case, n should be set to the size of the corpus, so we select a complete ranking.

Third, we use $\mathcal{R}$ to indicate a specification of general relevance judgment information. For most metrics this will simply be the set of documents that are relevant to query. though in the general case it could presumably be arbitrarily complex. Instance recall, for example, uses multiple separate relevance sets, and user-oriented metrics may incorporate user state information in $\mathcal{R}$.

Intuitively, the EMP captures the notion that if we believe in the evaluative power of a particular metric, we should guide the probabilistic model to directly optimize for said metric. Though a simple statement, equation 5.1 results in a variety of compelling objective functions when applied to specific metrics. The remainder of this chapter focuses on deriving usable objective functions for the canonical precision and recall metrics, the newly defined 1-call and n-call metrics from the previous chapter, and the reciprocal rank and search length metrics. We also briefly touch on derivations for

some other commonly used metrics, showing the wide applicability of our approach.

5.1 Notation and Common Mathematics

Let us briefly review the relevant notation. We will continue with the same notation from previous chapters, though we will find it useful to define some additional variables for notational consistency with the PRP formulation. As before, d_1 through d_n represent the documents at each rank of a result set, and D_n represents the set of those top n documents. We will now also use r_i as a binary random variable indicating the relevance of the ith document in the ranking. In particular, note that

$$E[I_R(d_i)] = E[r_i \mid d_i]. \tag{5.2}$$

More generally, we can replace any combination of $I_R(d_1)$ through $I_R(d_n)$:

$$E[f(I_R(d_1), \ldots, I_R(d_n))] = E[f(r_1, \ldots, r_n) \mid D_n]. \tag{5.3}$$

Additionally, recall that the expectation of a binary variable is simply its probability:

$$E[r_i \mid d_i] = \Pr[r_i \mid d_i]. \tag{5.4}$$

5.2 Precision and Recall

We first consider the standard metrics, precision at n and recall at n, and show that EMP suggests an objective function that is the same as the one maximized by PRP, so that PRP is optimal for these metrics. That PRP is optimal for precision and recall, was previously known [22].

Recall that precision at n is simply the number of relevant documents retrieved in the top n, divided by n. We can derive a specific ranking equation for precision by

finding the expectation of its mathematical expression, equation 4.1.

$$\begin{aligned} E[P_n(D_n, R) \mid q] &= E\left[\frac{\sum_{i=1}^{n} I_R(d_i)}{n} \,\middle|\, q\right] \\ &= E\left[\frac{\sum_{i=1}^{n} r_i}{n} \,\middle|\, q, D_n\right] \\ &\stackrel{\text{mt}}{=} \sum_{i=1}^{n} E[r_i \mid q, D_n] \\ &= \sum_{i=1}^{n} \Pr[r_i \mid q, D_n] \\ &= \sum_{i=1}^{n} \Pr[r_i \mid q, d_i] \end{aligned} \tag{5.5}$$

This derivation indicates that EMP should choose the result set that maximizes the sum of the conditionally independent probabilities of relevance of the top n results. Intuitively, this means choosing the n distinct most likely documents to be relevant. PRP ranks by choosing the document with maximum $\Pr[r_i|q, d_i]$ for each valid i successively. Therefore PRP implicitly optimizes the sum of those quantities for the top n documents (for any n), which is exactly equation 5.5. This derivation shows that EMP reduces to PRP ranking for the case of precision at n.

Intuitively, the same argument works for showing that PRP is optimal (in the EMP framework) for recall at n, or $R_n(D, R)$. However, note that the denominator of the expression for recall involves a term for the number of relevant documents, which is itself an unknown random variable. To be rigorous, the calculation should consider the number of relevant documents stochastically. Let X be a random variable indicating the total number of relevant documents for this query, and assume that at

least one such document exists (i.e., $\Pr[X = 0] = 0$). Then:

$$\begin{aligned}
&E[R_n(D_n, R) \mid q] \\
&= E\left[\frac{\sum_{i=1}^{n} I_R(d_i)}{X} \,\middle|\, q\right] \\
&= E\left[\frac{\sum_{i=1}^{n} r_i}{X} \,\middle|\, q, D_n\right] \\
&= \sum_{i=1}^{n} E\left[\frac{r_i}{X} \,\middle|\, q, d_i\right]
\end{aligned}$$

Partitioning by r_i and $\neg r_i$, and noting the latter case goes to zero:

$$= \sum_{i=1}^{n} E\left[\frac{1}{X} \,\middle|\, q, d_i, r_i\right] \Pr[r_i \mid q, d_i]$$

The value of $E\left[\frac{1}{X} \mid q, d_i, r_i\right]$ is always the same regardless of i, so we can drop it as a constant and reduce the ranking to:

$$\stackrel{\text{mt}}{=} \sum_{i=1}^{n} \Pr[r_i \mid q, d_i] \tag{5.6}$$

This final ranking is the same as the ranking expression for precision, equation 5.5, and so by the same argument that we used with precision, PRP is optimal for recall at n.

Interestingly, if we had reversed the PRP ranking within the top n, the result set would still be optimal according to EMP. This is a desired effect—by definition, precision and recall at n do not care about how the documents are ordered within the top n, so neither should the algorithms that optimize for those metrics. The PRP ranking happens to be one possible optimal solution for EMP on precision and recall.

5.2.1 Other Metrics Derived from Precision and Recall

We had introduced a number of metrics that were essentially variations on precision and recall, including R-precision, average precision, F-measure, and E-measure. We

will find that for all but one of these metrics, PRP produces an optimal result set in the EMP sense.

R-precision, the precision at rank equal to the number of relevant documents, presents an interesting challenge, in that we do not know where the cutoff rank is. Hence, we have to assume a full ranking of all m corpus documents. The formal derivation is a bit more complex, and requires conditioning on possible values of X from one to the size of the corpus:

$$\begin{aligned}
&E[RP(D_m, R) \mid q] \\
&= E\left[\left.\frac{\sum_{i=1}^{X} I_R(d_i)}{X}\right| q\right] \\
&= E\left[\left.\frac{\sum_{i=1}^{X} r_i}{X}\right| q, D_n\right] \\
&= \sum_{x=1}^{m} \frac{\Pr[X = x \mid q, D_n]}{x} \sum_{i=1}^{x} \Pr[r_i \mid q, D_n, X = x]
\end{aligned}$$

Under our probabilistic model, $\Pr[r_i \mid q, D_n, X = x] = \Pr[r_i \mid q, D_n]$ (that is, the probability of relevance of a particular document is independent of the total number of relevant documents) because each r_i is assumed independent given a document, and $X = x$ is merely a constraint on the sum of the r_i's:

$$\begin{aligned}
&= \sum_{x=1}^{m} \frac{\Pr[X = x \mid q, D_n]}{x} \sum_{i=1}^{x} \Pr[r_i \mid q, D_n] \\
&= \sum_{i=1}^{m} \Pr[r_i \mid q, D_n] \left(\sum_{x=i}^{m} \frac{\Pr[X = x \mid q, D_n]}{x} \right) \qquad (5.7)
\end{aligned}$$

We can view equation 5.7 as a weighted sum of the terms $\Pr[r_i \mid q, D_n]$ for $1 < i < m$, with higher weights for earlier i (because the coefficient's sum includes more terms for smaller i). This suggests that the way to optimize this expression would be to select the highest probability document for the first position, with its highest weight, the second highest document for the second, and so on. This is exactly PRP ranking, we we have found that EMP on R-precision once again reduces to PRP ranking, over

the entire corpus. A subtle but notable distinction between the EMP derivations for precision/recall at n and R-precision is that the former is agnostic towards the ordering within the top n, whereas the latter *requires* that documents are ordered exactly by relevance probability. In other words, PRP is just one optimal ranking for precision/recall at n, but is the only optimal ranking for R-precision.

An argument similar to the one for recall shows that E-measure (which is itself a generalization of F-measure) is also optimized by PRP in the EMP framework. This is again not a surprising result—a result set that is optimal for both precision and recall, should be optimal for any strictly increasing function of the two metrics. E-measure, which is a weighted harmonic mean of precision and recall, is such an increasing function.

We will return to our final precision/recall variant, average precision, at the end of this chapter. Perhaps surprisingly, average precision does not reduce to PRP ranking under the EMP framework. The derivation is more complex and will be clearer after examining EMP derivations for other metrics.

5.3 1-call

We now our turn our attention to the k-call at n metrics, which will be our first exposure to the EMP's ability to suggest novel ranking algorithms. We begin by examining the simplest case, $k = 1$. Using the logical form of the metric (equation 4.13), maximizing the expectation is equivalent to maximizing the probability that the logical statement is true:

$$
\begin{aligned}
& E[C_{n,1}(D_n, R) \mid q] \\
&= \Pr\left[\bigcup_{i=1}^{n} I_R(d_i) \,\middle|\, q\right] \\
&= \Pr\left[\bigcup_{i=1}^{n} r_i \,\middle|\, q, D_n\right]
\end{aligned}
$$

Using the law of total probability (equation A.2), partitioning on the events r_1 and $\neg r_1$:

$$= \Pr[r_1 \mid q, D_n] + \Pr[\neg r_1 \mid q, D_n] \Pr\left[\bigcup_{i=2}^{n} r_i \,\middle|\, q, D_n, \neg r_1\right]$$

Note that conditioned on r_1, the 1-call criterion is satisfied, so the first term is just the probability of r_1. Continuing this partitioning for $i = 2$ to n:

$$\begin{aligned}
&= \Pr[r_1 \mid q, D_n] + \Pr[\neg r_1 \mid q, D_n] \times \\
&\quad \left(\Pr[r_2 \mid q, D_n, \neg r_1] + \Pr[\neg r_2 \mid q, D_n, \neg r_1] \Pr\left[\bigcup_{i=3}^{n} r_i \,\middle|\, q, D_n, \neg r_1, \neg r_2\right] \right) \\
&= \Pr[r_1 \mid q, d_1] + \Pr[\neg r_1 \mid q, d_1](\Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots(\Pr[r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] + \Pr[\neg r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] \times \\
&\quad \Pr[r_n \mid q, D_n, \neg r_1, \ldots, \neg r_{n-1}])))
\end{aligned} \tag{5.8}$$

This is the first time that the EMP has suggested a different ranking mechanism than standard PRP. To optimize 1-call, one should evaluate this expression for each document set of size n, and output the document set with the highest score (probability) as the results.[1]

5.3.1 Computing EMP Probabilities

How do we actually go about evaluating an expression such as equation 5.8, with its complex conditional probabilities of relevance? For simple terms such as $\Pr[r_1 \mid q, d_1]$, we just apply Bayes' Rule and obtain a form in terms of relevant and irrelevant distributions as we saw in chapter 3:

$$\Pr[r_1 \mid q, d_1] = \frac{\Pr[d_1 \mid q, r_1] \Pr[r_1 \mid q]}{\Pr[d_1 \mid q, r_1] \Pr[r_1 \mid q] + \Pr[d_1 \mid \neg q, r_1] \Pr[\neg r_1 \mid q]} \tag{5.9}$$

[1] An alternative derivation would be to minimize the negative of the probability of 1-call, or $\Pr\left[\bigcap_{i=1}^{n} \neg r_i \mid q, D_n\right]$, which will lead to an equivalent ranking formulation in a different algebraic form.

Unlike before, however, we cannot simply drop the terms $\Pr[r_1 \mid q]$ and $\Pr[\neg r_1 \mid q]$, because the complete ranking expression of equation 5.8 involves addition. To be more flexible, we could assign a prior to the unconditional probability of relevance. For simplicity, however, we will treat these terms as tunable constants.

Equation 5.8 also involves more complex terms, such as $\Pr[r_2 \mid q, D_2, \neg r_1]$. To compute these terms, we apply Bayes' Rule as before, and use the relevance variables in the condition (in this case, $\neg r_1$ and d_1) to *feed back* documents as training examples to the corresponding models. Using this term as an example:

$$\Pr[r_2 \mid q, D_2, \neg r_1] = \frac{\Pr[d_2 \mid \neg q, d_1, \neg r_1, r_2] \Pr[r_2 \mid q, d_1, \neg r_1]}{\Pr[d_2 \mid \neg q, d_1, \neg r_1, r_2] \Pr[r_2 \mid q, d_1, \neg r_1] + \Pr[d_2 \mid \neg q, d_1, \neg r_1, \neg r_2] \Pr[\neg r_2 \mid q, d_1, \neg r_1]} \tag{5.10}$$

In this formulation, we first assume that the prior probability of relevance terms, $\Pr[r_2 \mid q, d_1, \neg r_1]$ and $\Pr[\neg r_2 \mid q, d_1, \neg r_1]$, are the same constant $\Pr[r]$ as we used earlier. In our simplified model, the r_i are independent unless knowledge of the previous relevant or irrelevant document is part of the condition.[2] The probabilities of documents are then computed as before, using the multinomial model with a Dirichlet prior. Unlike before, we update the distributions with the information in the conditioning. In this case, the condition tells us both d_1 and $\neg r_1$—that is, what the first document is, and that it should be considered irrelevant. Just like we trained the relevant distribution assuming that the query q was a sample, we now can use d_1 as a sample from the irrelevant distribution, and use d_1 as a piece of training data, using the MAP estimator. Thus, the document terms such as $\Pr[d_2 \mid \neg q, d_1, \neg r_1, r_2]$ can be computed on the basis of the updated distributions. Each term of the complete ranking expression can be computed similarly; for each pair of terms r_i and d_i in the condition, we incorporate d_i as a training example into the relevant document distribution, and for each pair $\neg r_i$ and d_i, we incorporate d_i as a irrelevant docu-

[2] In fact, this point will become entirely moot for greedy heuristic optimizations of this EMP objective, as we will see in the next chapter.

ment training example. In general, we can compute arbitrarily complex conditional probabilities, by incorporating each document marked as relevant or irrelevant as training data into the respective distributions. Returning to equation 5.8, this procedure allows us to compute each of the probabilities, by using a succession of irrelevant distributions that are trained on all previous documents of the result set.

As a trivial example of how this ranking algorithm is *not* the same as PRP, consider a one-word query "foo" and three corpus documents (1) "foo foo foo"; (2) "foo foo"; and (3) "foo baz." For a result set of size two, a PRP ranking would select the similar documents (1) and (2) because both have the most occurrences of the query term, but (depending on the precise model weightings used for training) a 1-call optimizer would select documents (1) and (3), as the word "foo" gets added to the irrelevant distribution after document (1).

5.4 n-call

The EMP formulation for n-call is very similar to that for 1-call. We again use the logical form of the metric from equation 4.14:

$$\begin{aligned} E[C_{n,n}(D_n, R) \mid q] &= \Pr\left[\bigcap_{i=1}^{n} I_R(d_i) \,\middle|\, q\right] \\ &= \Pr\left[\bigcap_{i=1}^{n} r_i \,\middle|\, q, D_n\right] \\ &= \prod_{i=1}^{n} \Pr[r_i \mid q, D_i, r_1, \ldots, r_{i-1}] \end{aligned} \tag{5.11}$$

In the final step, we factorized a conjunction in the standard way using chained conditional probabilities. As before, the individual conditional probabilities are computed based on the relevance and irrelevance models with feedback documents. In this case, the multiplicand $\Pr[r_i \mid q, D_i, r_1, \ldots, r_{i-1}]$ for any i is the probability of relevance of the ith document, where the relevant distribution is updated to include all the previous documents in the ranking, and the irrelevant distribution is unchanged.

5.5 k-call for Intermediate k

The case of general k-call is significantly more complex. We have to sum over the probability of all conforming *configurations*. A configuration is a precise assignment of relevance and irrelevance to each of the top n documents. From a logical standpoint, a configuration is just a conjunction of relevance variables, such as $r_1 \cap \neg r_2 \cap \neg r_3 \cap r_4 \cap \ldots r_n$. Let $|C|$ indicate the size of the configuration (number of variables), and $\lfloor C \rfloor$ indicate the number of positive (relevant) variables. Then we could represent general k-call as:

$$E[C_{n,k}(D_n, R) \mid q] = \sum_{\substack{\lfloor C \rfloor \geq k \\ |C|=n}} \Pr[C \mid q, D_n] \tag{5.12}$$

This expression is a brute force sum over all possible result set relevance configurations that have at least k relevant. Sometimes it is easier to subtract from one the sum over all nonconforming configurations, for low values of k.

Depending on the precise approximation heuristic used (see chapter 6), we may be able to simplify the ranking expression, but can still be difficult to compute for large n and k near $n/2$.

5.6 Reciprocal Rank and Search Length

Let us now apply the EMP ranking approach to the SL and RR metrics, whose derivations are rather similar. In both cases, we apply fairly straightforward simplification to the algebraic forms of the metrics introduced in chapter 4.

First, for reciprocal rank, equation 4.7:

$$
\begin{aligned}
&E[RR_n(D_n, R) \mid q] \\
&= E\left[I_R(d_1) + \sum_{i=2}^{n} \frac{I_R(d_i)}{i} \prod_{j=1}^{i-1} (1 - I_R(d_j)) \,\middle|\, q \right] \\
&= E\left[r_1 + \sum_{i=2}^{n} \frac{r_i}{i} \prod_{j=1}^{i-1} (1 - r_j) \,\middle|\, q, D_n \right] \\
&= \Pr[r_1 \mid q, d_1] + \sum_{i=2}^{n} \frac{1}{i} E\left[r_i \prod_{j=1}^{i-1} (1 - r_j) \,\middle|\, q, D_n \right]
\end{aligned}
$$

Repeatedly applying the law of total expectation on partitions r_1 and $\neg r_1$ through r_n and $\neg r_n$ on the expectation:

$$
\begin{aligned}
&= \Pr[r_1 \mid q, d_1] + \\
&\quad \sum_{i=2}^{n} \frac{1}{i} \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}]
\end{aligned}
\tag{5.13}
$$

Now, for search length, equation 4.8:

$$
\begin{aligned}
&E[SL_n(D_n, R) \mid q] \\
&= E\left[\sum_{i=2}^{n} (i-1) I_R(d_i) \prod_{j=1}^{i-1} (1 - I_R(d_j)) + n \prod_{i=1}^{n} (1 - I_R(d_i)) \,\middle|\, q \right] \\
&= E\left[\sum_{i=2}^{n} (i-1) r_i \prod_{j=1}^{i-1} (1 - r_j) + n \prod_{i=1}^{n} (1 - r_i) \,\middle|\, q, D_n \right] \\
&= \sum_{i=2}^{n} (i-1) E\left[r_i \prod_{j=1}^{i-1} (1 - r_j) \,\middle|\, q, D_n \right] + nE\left[\prod_{i=1}^{n} (1 - r_i) \,\middle|\, q, D_n \right]
\end{aligned}
$$

Repeatedly applying the law of total expectation on partitions r_1 and $\neg r_1$ through r_n and $\neg r_n$ for both expectations:

$$= \sum_{i=2}^{n} (i-1) \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}]$$
$$+ n \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_n \mid q, D_n, \neg r_1, \ldots, \neg r_{n-1}] \quad (5.14)$$

Not surprisingly, the reciprocal rank and search length metrics appear to be optimized similarly—equations 5.13 and 5.14 share similar chained products of relevance probabilities. In both cases, the EMP ranking is a natural representation of the metric in a probabilistic setting. As we will observe in the next chapter, the similarity of the two algorithms means that they share a common heuristically optimizing algorithm.

5.7 Instance Recall

Instance recall is different from the other metrics because there is no single set of relevant documents that defines the metric. Rather, there are multiple such relevant sets. It will turn out, however, that instance recall is optimized in the same way as 1-call. The intuition behind this connection is that instance recall is in some sense 1-call over k instances. Each instance is retrieved (and instance recall score increased by $1/k$) if the 1-call criterion for that instance's relevant set is satisfied.

Mathematically, the picture is slightly more complex than previous metrics due to multiple relevant sets. We will expand our notation to also include relevance from the different relevant sets. Let r_i^j be the indicator of relevance of the ith document,

d_i, for the jth instance, for $0 < j < k$. Then:

$$
\begin{aligned}
&E[IR_{n,k}(D_n, R_1, \ldots, R_k) \mid q] \\
&= E\left[\frac{\sum_{j=1}^{k}\left(1 - \prod_{i=1}^{n}(1 - I_{R_j}(d_i))\right)}{k} \,\middle|\, q\right] \\
&= E\left[\frac{\sum_{j=1}^{k}\left(1 - \prod_{i=1}^{n}(1 - r_i^j)\right)}{k} \,\middle|\, q, D_n\right] \\
&= \frac{1}{k}\sum_{j=1}^{k} E\left[1 - \prod_{i=1}^{n}(1 - r_i^j) \,\middle|\, q, D_n\right]
\end{aligned}
$$

We now make a key assumption—that the priors for each of the relevant and irrelevant distributions for each instance are the same. Given this assumption, each term of the inner sum is actually the same, because the trained models are identical. Let r_i be an aggregate variable indicating the common calculations for d_i.

$$
\begin{aligned}
&= \frac{1}{k}\sum_{j=1}^{k} E\left[1 - \prod_{i=1}^{n}(1 - r_i) \,\middle|\, q, D_n\right] \\
&= E\left[1 - \prod_{i=1}^{n}(1 - r_i) \,\middle|\, q, D_n\right]
\end{aligned}
$$

This expression is the same as the 1-call metric, so we use equation 5.8:

$$
\begin{aligned}
&= \Pr[r_1 \mid q, d_1] + \Pr[\neg r_1 \mid q, D_n](\Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots (Pr[r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] + \Pr[\neg r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] \times \\
&\quad \Pr[\neg r_n \mid q, D_n, \neg r_1, \ldots, \neg r_{n-1}])))
\end{aligned} \tag{5.15}
$$

We had made the assumption that the generative model of each instance was the same. This assumption, though strong, is the best we can make given no other differentiating information about what the instances look like.

5.8 Mean Average Precision

We conclude this chapter by returning to the average precision metric. Once again, let X be a random variable indicating the total number of relevant documents. Let us work through this derivation:

$$
\begin{aligned}
&E[AP_n(D_n, R) \mid q] \\
&= E\left[\frac{\sum_{i=1}^{n} I_R(d_i) \sum_{j=1}^{i} I_R(d_j)/i}{X} \,\middle|\, q\right] \\
&= E\left[\frac{\sum_{i=1}^{n} r_i \sum_{j=1}^{i} r_j/i}{X} \,\middle|\, q, D_n\right] \\
&= \sum_{x=1}^{m} \frac{\Pr[X = x \mid q, D_n]}{x} \sum_{i=1}^{n} E\left[r_i \sum_{j=1}^{i} r_j/i \,\middle|\, q, D_n, X = x\right] \\
&= \sum_{x=1}^{m} \frac{\Pr[X = x \mid q, D_n]}{x} \sum_{i=1}^{n} \frac{\Pr[r_i \mid q, d_i, X = x]}{i} \sum_{j=1}^{i-1} \Pr[r_j \mid q, d_i, d_j, r_i, X = x].
\end{aligned}
\tag{5.16}
$$

Knowing the number of relevant documents (that is, conditioning on a value of X) does not affect the choice of which D_n is optimal. Hence we can drop the outermost sum, and the coefficient $\Pr[X = x]/x$, to obtain:

$$
E[AP_n(D_n, R) \mid q] \stackrel{\text{mt}}{=} \sum_{i=1}^{n} \frac{\Pr[r_i \mid q, d_i]}{i} \sum_{j=1}^{i-1} \Pr[r_j \mid q, d_i, d_j, r_i]. \tag{5.17}
$$

This expression, though fairly complex, is computable for any given D_n, and provides a novel ranking mechanism for average precision.

5.9 Summary

We began this chapter with the simple observation that given a probabilistic model of relevance, it is possible to take an expected value of an arbitrary metric and use this expectation as a ranking value. We applied this general principle to a variety

of specific metrics, in some cases finding that the conventional probability ranking principle is optimal, and in other cases deriving new ranking values that suggest novel algorithms. In the next chapter we take a closer look at the novel ranking expressions, showing how we can reduce the complexity of calculating them by using some appropriate algorithmically-motivated heuristics.

Chapter 6

Algorithmic Approaches

In general, the PRP objective function can be optimized relatively efficiently by computing an independent score for each document. That is, we need to calculate $O(m)$ separate scores, where m is the corpus size. However, the objective functions suggested by EMP are more complex and do not factor simply as in PRP—there are conditionings between relevance variables. It is no longer possible to judge each document individually, so more complex optimization techniques are necessary.

Consider a fully general EMP-based objective function for an arbitrarily complex metric. Say that we want to retrieve n documents from a corpus of m documents. A perfect optimization would require calculating the expectation of the metric for all $P(m, n) = m^{\Theta(n)}$ distinct ordered subsets of documents.[1] This is obviously a combinatorially explosive number, and for sufficiently complex metrics each possible result set may have a score calculated completely independently of other result sets, even for result sets different by one result. Therefore, there are no efficient ways to always find the exact optimal result set in the general case.

However, there are three ways around this intractability. First, for certain specific metrics, it may be possible to exactly optimize without calculating a score for each possible result set. This is the case, for example, with precision and recall at n, which is optimized by PRP, and involves only calculating m scores (one independent

[1] $P(a, b)$ is the number of ordered permutations of size b that can be drawn from a collection of size a.

score for each document). Second, we may be able to approximate the expectation, such as by explicitly assuming independence between certain variables and ignoring conditionings. Third, we can only look at a portion of the search space, by taking approaches that heuristically cull possible result sets without calculating the actual score. Of course, all three approaches can be used in combination.

In this section, we focus primarily on the third approach. The first approach is very specific to the metric under consideration, and is not a "heuristic" *per se* because it implies that the optimization is still exact. The second approach also depends on the metric; some metrics are certainly more easily approximated than others. In contrast, taking the third approach allows us to explore some general approaches to reducing the search space that can be applied (with varying degrees of success) to any metric.

We will examine a straightforward greedy algorithm that selects documents successively in rank order, as well as a pruning algorithm that can be used to effectively reduce the corpus size (and turn the EMP problem into a reranking problem). We also touch on other possibilities, such as local search.

We focus specifically on the 1-call and n-call metrics introduced earlier. We show that for these two metrics, exact optimization is NP-hard and thus intractable. We then show how we can apply the greedy approach to simplify the ranking calculations for these metrics.

6.1 A Detour on Heuristics

It is important to clarify at this point what is exactly meant by a *heuristic*. Traditionally in IR, heuristics are modifications of the ranking/scoring algorithms that are found, empirically, to improve overall system relevance. Generally these heuristics are found and tested in an *ad hoc* manner, and require additional algorithmic complexity (and thus, space and time) to incorporate. These heuristics are an attempt to better represent a rather indefinite notion of relevance. In this thesis, we are using heuristic in an *algorithmic* sense—we start with a well defined notion of relevance, and

then compute a simpler, but inexact, approximation for tractability and efficiency purposes. If our model of relevance is "correct" (that is, reflective of what the user wants), then our heuristics actually should *hurt* relevance, because we are finding a suboptimal solution for the model optimization problem. Note that our usage of heuristic also clearly delineates the boundary between model and optimization—the theoretical model is orthogonal to the heuristic approaches we may take in approximation or search space reduction.

6.2 Greedy Algorithms

Greedy algorithms are commonly used as an algorithmic approach to many disparate problems. In general, a greedy algorithm incrementally builds a solution by making, at every step, the *locally* optimal choice, in the hope that these choices will lead to a *globally* optimal solution. For some problems, such as the *minimum spanning tree* problem in graph theory, natural greedy algorithms also give optimal solutions [7]. For some other problems, greedy algorithms give good approximate solutions; sometimes one can even prove bounds on the error of the greedy solution. Greedy approaches can be suboptimal because they imply that we do not consider all the possibilities in the search space—by committing ourselves to local optima early on, we may make choices that prevent us from reaching the global optimum later on.

Applying a greedy approach to EMP ranking, we can imagine selecting the documents for the result set *successively* rather than all at once. As we select each additional document, we try to maximize the expected value of the metric on the results selected *so far*, keeping the previously selected documents fixed. Thus, at each stage of ranking, the only degree of freedom is one document. In other words, in selecting the ith document of a ranking, we calculate the expected value of the metric given fixed values of d_1 through d_{i-1}, and considering every possible not-yet-selected document as d_i. Mathematically, we select d_i as:

$$d_i = \operatorname*{argmax}_{d \in C \setminus D_{i-1}} E[\mathcal{M}(D_n, \mathcal{R}) \mid q, D_{i-1}, d_i = d]. \tag{6.1}$$

While computing these expected values, we will encounter probability terms that depend on future documents in the ranking, that have yet to be selected. We have multiple alternatives for dealing with these terms. First, we could leave those terms in. In such a scenario, a probability of relevance term of future documents d_i, that is, $\Pr[r_i \mid \ldots]$, would be assumed to be constant. A term whose conditioning depends on a future document d_i, would be assumed to independent of that d_i, that is, $\Pr[\ldots \mid d_i] = \Pr[\ldots]$. Second, we could generalize the metric. Most of the metrics we have considered are a score at some rank position n. An easy generalization would be to optimize that same metric, but at the rank position of the document currently being selected. For example, for metric $\mathcal{M}$ at n, we would optimize $\mathcal{M}$ at 1 in selecting d_1, then $\mathcal{M}$ at 2 to select d_2, eventually reaching $\mathcal{M}$ at n for selecting the last document, d_n. Of course, the second approach is only feasible for metrics that have a specific cutoff. Third, another possibility is to start with n documents (selected by another algorithm, or at random) and replace greedily starting from this set—essentially a local search method that is described in greater detail later. In practice, for many metrics the first two approaches will often turn out to produce the same result.

The greedy algorithm requires calculating a score for each document at each rank position, about $\Theta(mn)$ scores, a significant reduction from the full optimization that requires $m^{\Theta(n)}$ calculations.

Note that this greedy approach can be applied for any metric—though we may expect that certain metrics perform better with a greedy approach than others. In particular, optimizing for precision and recall greedily is optimal, because a choice for the earlier positions in the ranking cannot be suboptimal at a later point in the ranking. For most metrics, we will not necessarily be so lucky—the greedy approach can only approximate the best solution.

6.3 Pruning

Another way to reduce the search space is to reduce the size of the corpus. A straightforward way to do this is to perform the search in two stages. First, we run a simple

ranking algorithm, such as PRP ranking. We then select the top m' documents from the first ranking, and use those as the restricted corpus for a second run with the full EMP ranking. In this manner, we could essentially reduce the number of score calculations from $P(m, n)$ to $m + P(m', n) = O(m + (m')^n)$, using PRP as the first ranking.

To ensure that we are not removing results we would otherwise select, the first ranking should be somewhat similar in its preferences to the second ranking. Of course, the first ranking should also be fast to compute. PRP is the most obvious first ranking, because the documents it selects towards the top would likely be considered reasonably "good," though not optimal, documents by any metric.

By performing pruning first, we are essentially turning our problem into one of reranking. Pruning can be used in conjunction with any of the other heuristics we have introduced, to effect further speedups.

6.4 Local Search

A more sophisticated algorithmic approach that we have began to explore is *local search*. In a generic sense, local search is the process of finding an optima in a discrete search space by moving from one state to a better "neighboring" state. In our problem, the states are possible result sets, and the scoring function over the states is simply the EMP ranking value. There are a variety of ways to define a state's neighborhood—a simple approach would be to have each pair of result sets with one result in one position different, be neighbors. In this case, exploring the search space with local search would just be swapping results one at a time until no result can be swapped in that will further improve the result set.

The greedy algorithms previously described can be considered special cases of local search where the search space includes partial result sets. The two approaches could also be combined, whereby we find a "seed" result set with a greedy algorithm and then further improve it with local search.

We have conducted some preliminary work with local search, and consider this

class of approaches an interesting direction of future work.

6.5 A Greedy 1-call Algorithm

The previous discussion examined search algorithms in the abstract. At this point, we turn our attention to how these different heuristics specifically affect EMP rankings for particular metrics. We first look at applying the approaches discussed above to simplifying the ranking function for 1-call. We show that exactly optimizing the 1-call objective is NP-hard by reducing the independent set problem to the optimal result set problem. We then look at how when we apply the greedy algorithm, we can greatly simplify the ranking expression.

6.5.1 Hardness

In this section we provide a proof of the "difficulty" of exactly optimizing the 1-call objective, by showing that the graph-theoretic independent set problem is polynomial time reducible to finding the optimal result set for 1-call. Because the MIS problem is NP-hard [11], such a proof indicates that optimal 1-call is also NP-hard.

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. An independent set is a subset of the vertices that are not neighbors with each other. That is, for any two vertices of the subset, there does not exist an edge between them. The independent set problem asks whether, for a given integer k between 1 and $|V|$, there exists an independent set of size k in the G. This decision problem is a canonical NP-complete (and thus NP-hard) problem.

Assume that we had a black box that solved the 1-call optimization exactly. If we can show that we could design a polynomial time algorithm that reduces (transforms) the independent set problem to the 1-call optimization, then we have shown that 1-call optimization is NP-hard. The basic idea behind the reduction is to map the vertices to documents. Then, edges between vertices in the graph correspond to a shared vocabulary term between the corresponding documents. If we can find the maximum 1-call set in this corpus, with a specific query, then we can verify whether

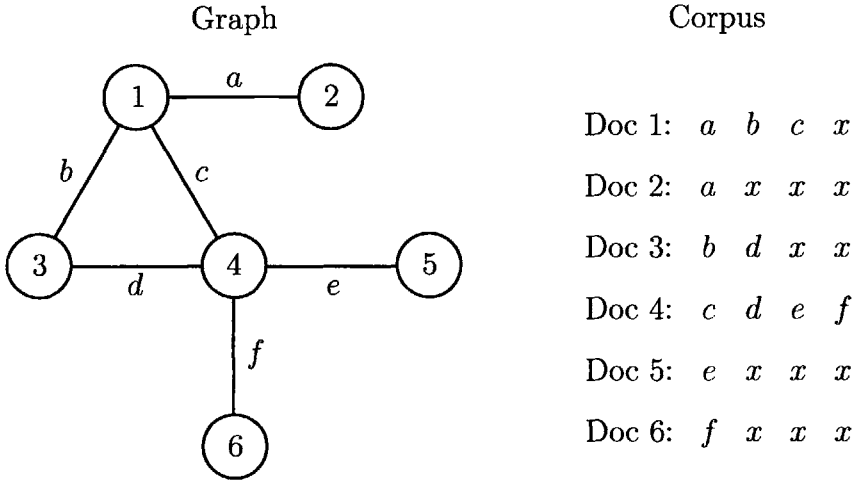

Figure 6-1: An example of the construction used in the NP-hardness proof. Each vertex corresponds to a document, and each edge corresponds to a term; documents are then padded to be of the same length.

that result set corresponds to an independent set in the original graph. If it does, then we answer "yes," and if not, then we answer "no." We rely on the fact that if there is an independent set of size k, then the best result set for 1-call at k set has to be an independent set.

Formally, let $G = (V, E)$ be a graph, with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\}$. Suppose we wish to check for the existence of an independent set of size k. We will create a corpus C with the following characteristics. For each vertex in v_i in V, create corresponding document d_i. For each edge $e_i = \{v_a, v_b\}$, create vocabulary term w_i and add it to documents d_a and d_b. Now pad all of the documents with additional words so that all documents are the same length. Specifically, let l be the highest degree of any vertex in the graph. Then for every document which has length less than l, add in new different words to fill the document up to length l. Each of these filler words, which we will call w'_{ij} for document d_i and for $1 \leq j \leq l$, is unique and only appears once in the corpus. For an example of this construction, see figure 6-1. In this figure, a graph is transformed into a corpus by the construction just described.

Using this setup, the reduction proceeds as follows. Given graph G and size k,

construct corpus C as described above. Run 1-call optimization at rank k on C. Check whether the result set corresponds to an independent set in G. If it does, return "yes," otherwise return "no."

First, the reduction is clearly polynomial time. Each document has length bounded by the number of vertices, and there are only as many documents as vertices.

We will prove that an independent set of size k exists in graph G, *if and only if* the result of 1-call at k optimization on corpus C will correspond to an independent set in graph G. The backward direction of this statement is vacuously true—if the result set corresponds to an independent set, then of course there exists an independent set. To prove the forward direction, assume that there exists an independent set of size k, but the result set from 1-call is not an independent set. Let the independent set be $I = \{v_{I_1}, \ldots, v_{I_k}\}$, and the result set be $D = \{d_{R_1}, \ldots, d_{R_k}\}$. In other words, I_1 through I_k are the indices of the vertices/documents that comprise the independent set, and R_1 through R_k are the indices of the vertices/documents that comprise the result set.

Recall the EMP ranking formula for 1-call, equation 5.8, reprinted here for convenience:

$$
\begin{aligned}
&E[C_{n,1}(D_n, R) \mid q] \\
&= \Pr[r_1 \mid q, d_1] + \Pr[\neg r_1 \mid q, d_1](\Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots(\Pr[r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] + \Pr[\neg r_{n-1} \mid q, D_{n-1}, \neg r_1, \ldots, \neg r_{n-2}] \times \\
&\quad \Pr[r_n \mid q, D_n, \neg r_1, \ldots, \neg r_{n-1}]))).
\end{aligned}
$$

We will compute this ranking formula for both the independent set of documents corresponding to I, and the 1-call optimization result D. An important observation to make about this ranking formula is that it is monotonically increasing as a function of $\Pr[r_1 \mid q, d_1]$, $\Pr[r_2 \mid q, D_2, \neg r_1]$, ..., $\Pr[r_n \mid q, D_n, \neg r_1, \ldots, \neg r_{n-1}]$. We will use this fact later on in the proof.

Assume that we use a Dirichlet prior with all hyperparameters equal to the con-

stant α for both the relevant and irrelevant distributions.[2] Let the prior probability of relevance ($\Pr[r_i]$ not conditioned on the respective d_i) be a constant p_r, and the size of the vocabulary be v. For simplicity, we can use an empty and noninformative query.[3] In this case, any independent set of size k will always have the same 1-call score, because the documents of such a set do not share terms, and are all of the same length l.

Returning to the ranking expression, we can apply Bayes' Rule to each term:

$$\begin{aligned}&\Pr[r_i \mid q, d_1 = d_{I_1}, \ldots, d_i = d_{I_i}, \neg r_1, \ldots, \neg r_{i-1}] \\ &= \left(1 + \frac{\Pr[d_i = d_{I_i} \mid \neg r_i, d_1 = d_{I_1}, \ldots, d_{i-1} = d_{I_{i-1}}, \neg r_1, \ldots, \neg r_{i-1}](1 - p_r)}{\Pr[d_i = d_{I_i} \mid r_i, d_1 = d_{I_1}, \ldots, d_{i-1} = d_{I_{i-1}}, \neg r_1, \ldots, \neg r_{i-1}]p_r}\right)^{-1}.\end{aligned} \tag{6.2}$$

To calculate these probabilities, we follow standard MAP training methodology, noting that d_{I_i} has no common words with previous documents because it is part of the independent set, and that no document has a word repeated. In particular, we feed back all the previous length l documents into the irrelevant distribution, so the resulting MAP estimate (equation 3.19) for the θ_j of a previously unseen term is:

$$\hat{\theta}_j = \frac{\alpha - 1}{(i-1)l + \alpha v - v}, \tag{6.3}$$

and the MAP estimate for any term of the relevant distribution (where there are no training examples):

$$\hat{\theta}_j = \frac{\alpha - 1}{\alpha v - v}, \tag{6.4}$$

Using these estimates, we proceed with equation 6.2 as follows:

$$\Pr[r_i \mid q, d_1 = d_{I_1}, \ldots, d_i = d_{I_i}, \neg r_1, \ldots, \neg r_{i-1}] = \left(1 + \frac{\left(\frac{\alpha-1}{(i-1)l+\alpha v-v}\right)^l (1 - p_r)}{\left(\frac{\alpha-1}{\alpha v-v}\right)^l p_r}\right)^{-1} \tag{6.5}$$

[2]The proof works equivalently for a prior with differentially set α's, but is algebraically complex.

[3]This proof works equally as well, with slightly more algebraic complexity, if we use a new term as the query and add once instance of it to all the documents.

Now consider the 1-call result set, which we assumed earlier was not independent. This implies that there exists at least two documents in this set that share a term. For each i from 1 to k, let s_i be the number of edges/terms shared with previous documents (that is, one of d_{R_1} through $d_{R_{i-1}}$). Those shared terms cannot have been previously encountered, because all shared terms are unique to a single edge, and thus a single pair of documents. As before with equation 6.2, we have:

$$\begin{aligned}
&\Pr[r_i \mid q, d_1 = d_{R_1}, \ldots, d_i = d_{R_i}, \neg r_1, \ldots, \neg r_{i-1}] \\
&= \left(1 + \frac{\Pr[d_i = d_{R_i} \mid \neg r_i, d_1 = d_{R_1}, \ldots, d_{i-1} = d_{R_{i-1}}, \neg r_1, \ldots, \neg r_{i-1}](1 - p_r)}{\Pr[d_i = d_{R_i} \mid r_i, d_1 = d_{R_1}, \ldots, d_{i-1} = d_{R_{i-1}}, \neg r_1, \ldots, \neg r_{i-1}]p_r}\right)^{-1}.
\end{aligned} \tag{6.6}$$

The MAP parameter estimate for the relevant distribution in the denominator is the same as with the independent set, equation 6.4. However, the ith document of the ranking could potentially duplicate previously seen terms that were fed back as training data, and so the MAP estimate for the irrelevant distribution would be readjusted:

$$\hat{\theta}_j = \frac{\alpha}{(i-1)l + \alpha v - v}, \tag{6.7}$$

We add the 1 because that is the number of observations of the shared term in the training data. Only s_i of the terms are shared, so only s_i of the parameter estimates follow equation 6.7; the rest follow equation 6.3, the same as with the independent set when there are no shared terms. Putting together these estimates, we obtain:

$$\begin{aligned}
&\Pr[r_i \mid q, d_1 = d_{R_1}, \ldots, d_i = d_{R_i}, \neg r_1, \ldots, \neg r_{i-1}] \\
&= \left(1 + \frac{\left(\frac{\alpha}{(i-1)l+\alpha v-v}\right)^{s_i}\left(\frac{\alpha-1}{(i-1)l+\alpha v-v}\right)^{l-s_i}(1 - p_r)}{\left(\frac{\alpha-1}{\alpha v-v}\right)^{l} p_r}\right)^{-1}.
\end{aligned} \tag{6.8}$$

When $s_i > 0$, the numerator of equation 6.8 is greater than the numerator of equation 6.5 (the probabilities for the independent set), and consequently the overall probability is lower for the result set than for the independent set.

We know that at least one of the s_i's for $1 < i < k$ must be greater than zero, because the result set is not an independent set; thus, at least one of the probabilities $\Pr[r_i \mid q, d_1 = d_{R_1}, \ldots, d_i = d_{R_i}, \neg r_1, \ldots, \neg r_{i-1}]$ must be smaller than the corresponding $\Pr[r_i \mid q, d_1 = d_{I_1}, \ldots, d_i = d_{I_i}, \neg r_1, \ldots, \neg r_{i-1}]$. Recalling our earlier remark that the total 1-call score is an increasing function of these probabilities for $1 < i < k$, we conclude that the independent set's 1-call score is higher than the result set's. But the result set is the optimal 1-call set, so we have derived a contradiction. Therefore we can finally conclude that if an independent set of size k exists in G, then the result set returned by the 1-call at k optimizer must be an independent set, and so the reduction is correct and 1-call optimization is NP-hard.

6.5.2 Simplifying the Ranking Formula

Now that we have shown the regular EMP ranking expression for 1-call is intractable to optimize, we turn our attention to applying the greedy algorithm to 1-call and n-call.

The greedy approach would simply be to select each document d_i successively. Recall that we had at least two approaches to deal with "future" documents, documents d_j for $j > i$. First, we could treat future document terms as constants. Alternatively, we could optimize the metric $C_{i,1}(D_i, R)$ at each rank i instead—cutting off the metric at the last available document. It turns out that these approaches are equivalent for 1-call, so we will take the second approach, which is more intuitive.

Because the only degree of freedom at each step of the greedy algorithm is the current document d_i, we can repeatedly apply monotonic transformations to simplify the ranking equation, alternating between dropping the head addend and the head

multiplicand, neither of which include the term d_i:

$$
\begin{aligned}
&E[C_{i,1}(D_i, R) \mid q] \\
&= \Pr[r_1 \mid q, d_1] + \Pr[\neg r_1 \mid q, d_1](\Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots(\Pr[r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] + \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \times \\
&\quad \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}])))) \\
&\stackrel{\text{mt}}{=} \Pr[\neg r_1 \mid q, d_1](\Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots(\Pr[r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] + \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \times \\
&\quad \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}])))) \\
&\stackrel{\text{mt}}{=} \Pr[r_2 \mid q, D_2, \neg r_1] + \Pr[\neg r_2 \mid q, D_2, \neg r_1] \times \\
&\quad (\ldots(\Pr[r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] + \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \times \\
&\quad \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}])) \\
&\qquad\qquad \vdots \\
&\stackrel{\text{mt}}{=} \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}]. \qquad (6.9)
\end{aligned}
$$

Applying the greedy algorithm immensely simplified the original ranking expression. The resulting ranking equation 6.9 simply requires that at each rank i, we choose the highest probability document, *conditioned on all the previously selected documents being irrelevant.* This ranking mechanism makes intuitive sense–if a previous document were able to satisfy the user query (i.e., was relevant), then we would not care about what documents were displayed subsequently. Thus we try to select the best new document assuming that all previous documents were irrelevant. The effect of feeding back the previous documents as irrelevant examples is that each result is selected to be as different as possible from the previously selected documents.

This formula also fits nicely with the Bayesian information retrieval model: the assumption that the previous documents are irrelevant is incorporated in a straightforward fashion as an update to the probability distribution associated with the irrelevant documents; the relevance probabilities of new documents are then computed using that updated irrelevant document distribution. We only need to keep track

of one relevant and one irrelevant distribution at each step of the greedy ranking procedure.

For the remainder of the thesis, we will call the greedy 1-call algorithm the *1-greedy* algorithm.

6.6 A Greedy n-call Algorithm

At the opposite end of the k-call spectrum, we can optimize for n-call at n. We can use the same reduction construction as with 1-call, but this time we would reduce the *clique* problem—given a graph G and a size k, indicate whether is a group of k vertices that has edges between each of its members. This problem is also NP-hard [11] and corresponds to the n-call problem, the proof following essentially the same structure with different formulae.

We follow the same procedure as with 1-call in deriving a simplified greedy ranking algorithm, except now we optimize i-call at rank i:

$$\begin{aligned} & E[C_{i,i}(D_i, R) \mid q] \\ &= \prod_{j=1}^{i} \Pr[r_j \mid q, D_j, r_1, \ldots, r_{j-1}] \\ &\stackrel{\text{mt}}{=} \Pr[r_i \mid q, D_i, r_1, \ldots, r_{i-1}]. \end{aligned} \tag{6.10}$$

This monotonic transformation follows easily from dropping all the multipliers of the last term. Interestingly, the n-call greedy algorithm tells us to feed back the previously selected documents as *positive* examples, so that they get added to the relevant distribution. Contrast this with the 1-call case, where documents were fed back to the irrelevant distribution. We will call greedy n-call the *n-greedy* algorithm.

The technique of pseudorelevance feedback, discussed previously in section 2.4.2, bears striking similarity to this ranking procedure. In both cases, we use the top results returned from a "previous" search as examples of relevant documents. The difference is that in pseudorelevance feedback, we gather together the top results

returned from an initial search, and feed them back together one time for a second pass. In n-greedy, we incrementally add the selected documents one by one as training examples for the relevant distribution, as they become selected. The net effect of both approaches is to choose a result set that is more tightly focused on one interpretation of the query. In a sense, we are arriving at a theoretical justification for a technique that previously was designed in an *ad hoc* manner.

6.7 Greedy Algorithms for Other Metrics

We march onward in a fairly mechanical manner, applying the greedy approach to the MRR and MSL, metrics. As before, we will tweak the metric at each step to optimize the metrics up to the available rank. First, reciprocal rank:

$$
\begin{aligned}
&E[RR_i(D_i, R) \mid q] \\
&= \Pr[r_1 \mid q, d_1] + \\
&\quad \sum_{j=2}^{n} \frac{1}{j} \Pr[r_j \mid q, D_j, \neg r_1, \ldots, \neg r_{j-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{j-1} \mid q, D_{j-1}, \neg r_1, \ldots, \neg r_{j-2}].
\end{aligned}
$$

If $i = 1$:

$$
\stackrel{\text{mt}}{=} \Pr[r_1 \mid q, d_1].
$$

Or if $i > 1$:

$$
\begin{aligned}
&\stackrel{\text{mt}}{=} \sum_{j=2}^{i} \frac{1}{j} \Pr[r_j \mid q, D_j, \neg r_1, \ldots, \neg r_{j-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{j-1} \mid q, D_{j-1}, \neg r_1, \ldots, \neg r_{j-2}] \\
&\stackrel{\text{mt}}{=} \frac{1}{i} \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \\
&\stackrel{\text{mt}}{=} \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}].
\end{aligned}
$$

To summarize the $i = 1$ and $i > 1$ case together, our final ranking criterion is simply:

$$\Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}]. \tag{6.11}$$

Now, we apply the greedy ranking approach to search length:

$$\begin{aligned}
&E[SL_i(D_i, R) \mid q] \\
&= \sum_{j=2}^{i} (j-1) \Pr[r_j \mid q, D_j, \neg r_1, \ldots, \neg r_{j-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{j-1} \mid q, D_{j-1}, \neg r_1, \ldots, \neg r_{j-2}] \\
&\quad + i \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}].
\end{aligned}$$

If $i = 1$:

$$\begin{aligned}
&\overset{\text{mt}}{=} \Pr[\neg r_1 \mid q, d_1] \\
&\overset{\text{mt}}{=} -\Pr[r_1 \mid q, d_1].
\end{aligned}$$

Or if $i > 1$:

$$\begin{aligned}
&\overset{\text{mt}}{=} (i-1) \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \\
&\quad + i \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \\
&\overset{\text{mt}}{=} (i-1) \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \\
&\quad - i \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \\
&= -\Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}] \Pr[\neg r_1 \mid q, d_1] \cdots \Pr[\neg r_{i-1} \mid q, D_{i-1}, \neg r_1, \ldots, \neg r_{i-2}] \\
&\overset{\text{mt}}{=} -\Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}]
\end{aligned}$$

Recall that we wish to *minimize* search length, so equivalently we could maximize the negative of the above expression, and as with reciprocal rank aggregate the separate cases for $i = n$ and $i < n$ together:

$$\Pr[r_i \mid q, D_i, \neg r_1, \ldots, \neg r_{i-1}]. \tag{6.12}$$

Interestingly, we note that both ranking expressions, equations 6.11 and 6.12, are the exact same as that of 1-call's, equation 6.9, telling us that the 1-greedy algorithm as previously defined is also a natural heuristic optimization for search length and reciprocal rank. In both cases, a moment of thought yields an intuition for the algorithm. In selecting the ith document, if pick the first relevant document, then we "terminate the search" and establish the reciprocal rank/search length; further documents do not affect the value of the metric. Thus, we should choose documents given that previous documents are irrelevant—leading us back to the 1-greedy algorithm.

Finally, we return to the instance recall metric, which we had shown earlier to have the same ranking expression as 1-call. Obviously, the greedy algorithm would be the same for instance recall as well, namely the 1-greedy algorithm. By running 1-greedy, we are effectively optimizing for 1-call for each instance simultaneously.

It is important to note that although we have described a *heuristic* that is effective for four metrics, 1-call, search length, reciprocal rank, and instance recall, they are in fact distinct metrics, and it is conceivable that more sophisticated algorithms could lead to divergent results that optimize one at the expense of the others. On the other hand, it is also conceivable that since these metrics are closely related, there is a ranking that (in expectation) optimizes several or all of them simultaneously.

6.8 A Greedy MAP Algorithm

We conclude this chapter by examining some preliminary work on applying the greedy approach to the commonly used mean average precision (MAP) metric. Following the

math, we have:

$$
\begin{aligned}
& E[AP_i(D_i, R) \mid q] \\
& \stackrel{\text{mt}}{=} \sum_{j=1}^{i} \frac{\Pr[r_j \mid q, d_j]}{j} \sum_{k=1}^{j-1} \Pr[r_k \mid q, d_j, d_k, r_j] \\
& \stackrel{\text{mt}}{=} \Pr[r_i \mid q, d_i] \sum_{k=1}^{i-1} \Pr[r_k \mid q, d_i, d_k, r_i]
\end{aligned}
$$

In words: *select document d_i at step i such that the sum of the probabilities of the previously selected documents conditioned on d_i being relevant, times the probability of the document itself being relevant conditioned only on the query, is maximal.*

The MAP greedy algorithm is novel and does not correspond to a standard ranking procedure; we do not perform experiments with it in this thesis, but consider it an interesting direction of future work.

6.9 Summary

In this chapter we have made the EMP practical, by showing how optimization heuristics can be used to make ranking values tractably computable. Greedy approaches are a natural heuristic to use, and can be applied in a multitude of ways to all of the metrics. We focused specifically on 1-call and n-call and derived simple, straightforward ranking algorithms for them. Along the way we gave an NP-hardness proof of full EMP optimization for 1-call (and by extension n-call). MRR, MSL, and instance recall are fortuitously also naturally heuristically optimized by the 1-call greedy optimizer.

Chapter 7

Experimental Results

At this point, we have developed the necessary theoretical background, to discuss empirical results. We focus our discussion of results on the greedy algorithm for 1-call and n-call, which we call 1-greedy and n-greedy as before. We will consistently use result sets of size ten in these experiments, so $n = 10$. Our baseline retrieval algorithm ranks by the standard probability ranking principle approach. We will be using various corpora and queries from TREC, which we described previously in the evaluation chapter.

Each corpus was filtered for stopwords and stemmed with a Porter stemmer. In each case, we ran k-greedy over the top 100 results from PRP. (Generally we found that our algorithms would select from within the top 100 PRP results even when given a choice from the entire corpus.)

Because we did not rank the entire corpus in our results (as doing so would be prohibitively slow), we compute search length only over the first ten results. If there are no relevant results in the top ten positions, we assume a search length of ten. Similarly, we assume a reciprocal rank of zero if a relevant result is not found in the top ten.

We used the set of *ad hoc* topics from TREC-1, TREC-2, and TREC-3 to set the weight parameters of our model appropriately. Using the weights we found, we then ran experiments over the TREC 2004 robust track, TREC-6, 7, 8 interactive tracks, and TREC-4 and TREC-6 *ad hoc* tracks.

Table 7.1: Google results are reranked with both PRP and 1-call, to show that 1-call is best at identifying diverse results.

Query	Rank	Google	PRP	1-call
trojan horse	1	Trojan Horse Attacks	Trojan Horse Removal	Trojan Horse Removal
	2	Trojan Horse - Webopedia	Trojan Horse Removal	Trojan Horse Detector
	3	Symantec - Trojan Horse	Trojan Horse Detector	Trojan Horse Inn
	4	Symantec - Glossary	Symantec - Trojan Horse	Trojan Horses
	5	Trojan horse - Wikipedia	Symantec - Trojan Horse	Achilles in Trojan horse
	6	History of the Trojan War	Achilles in Trojan horse	Trojan Horse - Wikimedia
	7	Trojan Horse	Trojan Horse Inn	Gay Activism as Trojan Horse
	8	What is a computer virus?	Trojan Horses	Trojan Horse Removal
	9	CERT Advisory Trojan Horses	Trojan Horse - Wikimedia	Trojan Horse game expansion
	10	Trojan horse - Whatis.com	Trojan Horse Scandal	Acid Trojan Horse
virus	1	Symantec Corp.	McAfee Virus Definitions	McAfee Virus Definitions
	2	Symantec Virus Hoax Page	Anti Virus Directory	Anti Virus Directory
	3	Symantec Security Response	Virus Threat Center	50 latest virus descriptions
	4	Norton AntiVirus	Virus Threat Center	Virus Threat Center
	5	Trend Micro	50 latest virus descriptions	West Nile Virus
	6	McAfee	Vaccinia virus	Vaccinia virus
	7	Virus Bulletin	Symantec - FatCat Hoax	Summary virus table
	8	AVG Anti Virus	Symantec - Londhouse Hoax	alt.comp.virus FAQ
	9	Vmyths.com	West Nile Virus	Panda Software
	10	Sophos Anti-Virus	Symantec - Hairy Palms Hoax	Sophos virus analyses

7.1 Google Examples

To gain an understanding of how 1-call promotes diversity, we also ran PRP and 1-greedy over the top 1000 results returned by Google for two prototypically ambiguous queries, "trojan horse" and "virus." We used the titles, summaries, and snippets of Google's results to form a corpus of 1000 documents for each query.

The titles of the top 10 Google results, and the PRP and 1-call rerankings, are shown in table 7.1. (The titles have been shortened for fit.) Different table cell shadings indicate different broad topic interpretation of a result (e.g., white for computer viruses and gray for medical viruses). In the "trojan horse" example, 1-call returns a significantly more diverse set of results in the top 10 (spanning five distinct interpretations) than PRP and the original Google results, both of which only return two separate interpretations. The diversity for "virus" is also notable; 1-call returns the most medical (non-computing) virus results in the top ten, beating PRP. Interestingly, Google does not return any medical virus information in its top ten, so a user looking for that interpretation would be disappointed by Google.

7.2 Tuning the Weights

As with any model, our model has a set of tweakable weights that could greatly affect retrieval performance depending on how they are set. For our model, there are three key weights to consider. First, we have to consider how to weight the query's strength compared to the relevant distribution's prior. That is, how many query terms would need to be fed back to equal the relevant distribution's prior weight (that is, for the MAP parameter estimates to be equally influenced by either)? Second, we need to set the strength of the relevant distribution distribution prior with respect to the strength of the documents that we add to that distribution, for retrieval algorithms that rely on feedback documents into the relevant distribution such as n-greedy. In other words, this weight specifies how many documents would need to be fed back before the prior and the training documents have equal total strength. Third, we need to set the strength of the irrelevant distribution prior, for retrieval algorithms that rely on feedback into that distribution, such as 1-greedy. This weight is analogous to the previous weight, except for the irrelevant distribution. These weights are all independent; we first tune the query weight before tuning the other two.

To tune these weights, we used the corpus from the TREC-1, TREC-2, and TREC-3 *ad hoc* task, consisting of about 742,000 documents. There were 150 topics for these TRECs (topics 51 through 200).

Table 7.2: Tuning the weight of the query compared to the relevant distribution using TREC-1,2,3 (150 Topics). Bold is best.

Query weight	Precision at 10
1	0.009
1/5	0.113
1/10	0.191
1/50	0.388
1/100	0.451
1/500	**0.467**
1/1000	0.446

Let us focus on the query weight first. To make the results between PRP, 1-greedy, and 10-greedy most comparable, we should set the query weighting equivalently be-

tween the three algorithms, because the query weight is used analogously in each algorithm. Thus, we will only use PRP to tune the query weight, and simply use standard precision at ten to determine the best weight setting. Table 7.2 presents the results of this experiment. It appears that weighting the query as one five hundredth of the strength of the relevant distribution prior is optimal for our experimental setting. In practical terms, this means that the sum of the Dirichlet prior's hyperparameters should be five hundred times as large as the weight given to a single query.

We will be using both two query weightings for later experiments; first, one of the optimal weights, a prior weight of 500, and second, one of the nonoptimal ones, a prior weight of 50. The former case is the "strong" baseline, and the latter the "weak baseline." We are interested in both because we would like to see how our model improves performance in both the case of a bad starting model and a good starting model.

Now we look at how we set the relevant and irrelevant distribution priors. In the 1-greedy setting, a high irrelevant distribution prior weight smooths out the documents that are fed back, thus decreasing their importance. Note that in the extreme, if the the irrelevant distribution prior has a weight of infinity, our 1-greedy algorithm reduces to ranking by the probability ranking principle. In the 10-greedy setting, the strength of the relevant distribution prior is the critical factor—a low prior means that we tend to "lock in" faster to the top interpretations, at the cost of the smoothing effect of the background prior and the query itself.

Table 7.3 presents the 1-call, 10-call, MRR, and MSL results for the 1-greedy and 10-greedy algorithms using a variety of prior weightings, on the weak baseline, and table 7.4 on the strong baseline. On the weak baseline, we find that when the prior weight is well tuned, 1-greedy outperforms PRP on the metrics where it should—that is, 1-call at 10, MRR, and MSL. Similarly, with a well tuned prior weight, 10-greedy outperforms PRP on 10-call. On the strong baseline, the improvement for 1-call and 10-call are still noticeable for 1-greedy and 10-greedy, respectively; however, 1-greedy does not improve on MRR and MSL in the same way. Not surprisingly, when the baseline score is higher, it is more difficult to improve. Interesting, the optimal

Table 7.3: Tuning the weight of the relevant and irrelevant priors using TREC-1,2,3 (150 Topics), using the weak baseline (query weight of 1/50). Bold is best.

Metric	1-call at 10	10-call at 10	MRR	MSL	Prec. at 10
PRP	0.780	0.100	0.555	4.173	0.388
1-greedy (prior = 1)	**0.800**	0.020	**0.580**	**3.807**	0.336
1-greedy (prior = 10)	**0.800**	0.020	0.574	3.907	0.310
1-greedy (prior = 100)	0.767	0.013	0.561	4.073	0.281
1-greedy (prior = 1000)	0.733	0.007	0.538	4.593	0.210
10-greedy (prior = 1)	0.620	0.060	0.495	5.360	0.311
10-greedy (prior = 10)	0.673	0.140	0.519	4.893	0.433
10-greedy (prior = 100)	0.727	**0.167**	0.537	4.573	0.434
10-greedy (prior = 1000)	0.727	0.153	0.536	4.507	**0.435**

Table 7.4: Tuning the weight of the relevant and irrelevant priors using TREC-1,2,3 (150 Topics), using the strong baseline (query weight of 1/500). Bold is best.

Metric	1-call at 10	10-call at 10	MRR	MSL	Prec. at 10
PRP	0.867	0.120	**0.614**	3.267	0.467
1-greedy (prior = 1)	0.867	0.113	0.613	3.273	**0.471**
1-greedy (prior = 10)	0.873	0.120	**0.614**	3.267	0.466
1-greedy (prior = 100)	0.893	0.080	0.599	**3.253**	0.443
1-greedy (prior = 1000)	**0.900**	0.013	0.589	3.353	0.381
10-greedy (prior = 1)	0.700	0.073	0.547	4.587	0.370
10-greedy (prior = 10)	0.713	0.120	0.554	4.460	0.395
10-greedy (prior = 100)	0.727	0.127	0.554	4.400	0.412
10-greedy (prior = 1000)	0.740	**0.173**	0.543	4.513	0.452

settings for the priors are different depending on which baseline we run off of; this is likely because the weights actually do affect each other in unforeseen ways, which cannot be accounted for when tuning the weights independently as we are doing.

We conducted all experiments on the same machine under similar circumstances. Each PRP run, regardless of weight settings, took approximately an hour and 23 minutes. The 1-greedy and 10-greedy runs each took two hours and 30 minutes. Thanks to memoization in the implementation and judicious pruning, the 1-greedy and 10-greedy algorithms only required an additional 81% runtime.

Since TRECs 1, 2, and 3 were used for tuning weights, retrieval results on them were not meaningful. For further evaluation we applied 1-greedy and 10-greedy with the prior weight settings we found in this section, to a different corpus.

7.3 Robust Track Experiments

We turn our attention to the TREC 2004 robust track. The robust track uses a standard *ad hoc* retrieval framework, but is evaluated with an emphasis on the overall reliability of IR engines—that is, minimizing the number of queries for which the system performs badly. There were 249 topics in total[1], drawn from the *ad hoc* task of TREC-6,7,8 (topics 301 to 450), the 2003 robust track (topics 601-650), and the 2004 robust track (topics 651-700). The corpus consisted of about 528,000 documents. Note that there is no overlap between this corpus and the TREC-1,2,3 corpus, in either documents or topics.

From the 249 robust track topics, 50 were selected by TREC as being "difficult" queries for automatic search systems. We separately call out the results for these 50 topics. We used the weight settings found previously, with both the strong and weak baselines, and their respective

Table 7.5: Results on the robust track (249 topics) with the weak baseline. Bold is best.

All topics					
Method	1-call	10-call	MRR	MSL	P@10
PRP	0.791	0.020	0.563	3.052	0.333
1-greedy	**0.835**	0.004	**0.579**	**2.763**	0.269
10-greedy	0.671	**0.084**	0.517	3.992	**0.337**
50 difficult topics only					
Method	1-call	10-call	MRR	MSL	P@10
PRP	0.580	0.000	0.303	5.500	**0.160**
1-greedy	**0.620**	0.000	**0.333**	**5.000**	0.158
10-greedy	0.420	0.000	0.254	6.500	0.152

Table 7.5 presents the results for the robust track on the weak baseline, and table 7.6 shows the results for the strong baseline. In either case, we show a noticeable improvement in 1-call by using 1-greedy instead of PRP. When we restrict our attention to just the 50 difficult queries, the results overall are lower, but the improvement is more dramatic, particularly for the strong baseline. Similarly, 10-greedy's 10-call score is higher than the corresponding PRP and 1-greedy scores.

[1]One topic was dropped because the evaluators did not deem any documents relevant for it.

Table 7.6: Results on the robust track (249 topics) with the strong baseline. Bold is best.

All topics					
Method	1-call	10-call	MRR	MSL	P@10
PRP	0.863	0.020	**0.642**	**3.418**	**0.363**
1-greedy	**0.880**	0.004	0.624	3.578	0.284
10-greedy	0.711	**0.060**	0.590	4.402	0.354
50 difficult topics only					
Method	1-call	10-call	MRR	MSL	P@10
PRP	0.720	0.000	0.417	5.420	**0.196**
1-greedy	**0.780**	0.000	**0.422**	**5.000**	0.186
10-greedy	0.520	**0.020**	0.377	6.320	0.174

The difficult queries live up to their name for 10-call—none of our algorithms satisfy the strict 10-call criterion over that subset, except for 10-greedy with the strong baseline.

We also note that 1-greedy has worse precision at 10 than PRP. However, as we argued earlier, precision is not the appropriate metric for our task, so a lower precision score is not problematic. Indeed, a lower precision score is sometimes desirable, because the implicit goal of precision (many relevant documents) may be opposed to the implicit goal of 1-call (any relevant document).

Finally, our performance on the other metrics for which we are greedily optimizing, namely MRR and MSL, is better under 1-greedy than with PRP, with the weak baseline. Because our 1-greedy procedure attempts to diversify the result set after selecting the first result, we would expect that it would be more likely to find a relevant result for the next few positions than PRP (recall that both methods choose the first result identically). In other words, if the first result was not relevant, 1-greedy will be more likely to select something different, and thus, something relevant, for the second result. On the other hand, PRP will stick with the same interpretation of the query, so if the first document was of the wrong interpretation (and thus irrelevant) the second document would more likely continue that trend. To examine the gains we are making in MRR and MSL, consider figure 7-1, which graphs the location of the first relevant document for the topics. As the figure demonstrates, it is more often

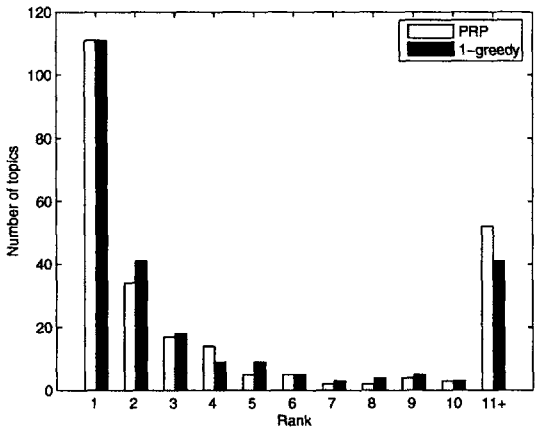

Figure 7-1: The location of the first relevant result for all of the robust track queries, using the weak baseline.

the case that 1-greedy chooses a relevant document for the second position.

Unfortunately, on the strong baseline we do not actually hurt MRR and MSL. Part of this could be due to nonoptimal weightings—in the previous section, we chose to use the weight that best improved 1-call, even though it had hurt MRR and MSL. A different weight setting would have at least maintained a constant level of these two metrics, but would have made for a smaller improvement in 1-call.

We conducted statistical significance tests on the robust track experiment's results to compare our greedy algorithms against the PRP baseline. First, consider the weak baseline. For 1-greedy vs. PRP on 1-call, a one-tailed McNemar test gives $p = 0.026$, which indicates significance at the 5% level. For 10-greedy vs. PRP on 10-call, $p = 0.0002$, which indicates significance at the 1% level. Using a one-tailed Wilcoxon test, we find that for 1-greedy vs. PRP on MRR and MSL, $p = 0.314$, which is not statistically significant. On the strong baseline, the results are generally not as significant, because the improvements are smaller over the same number of queries.

7.4 Instance Retrieval Experiments

As described in chapter 4, optimizing for 1-call has the side-effect of seeking diversity in the result set—it returns more distinct interpretations of the query in expectation. The TREC-6, 7, and 8 interactive track runs afford us a unique opportunity to test

the performance of our system for diversity, because each run's topics were annotated with multiple "instances" (i.e., subtopics) that described its different facets [16]. For example, topic 392i's description reads:

What are the applications of robotics in the world today?

The annotators identified 35 different subtopics, describing 35 different robotics applications. The document judgments were also annotated with the instances that they covered. In total, there were 20 topics, with between 7 and 56 aspects each, and a corpus of about 210,000 documents.

Table 7.7: Results for instance recall on the interactive track (20 Topics).

Method	Instance recall at 10
PRP	0.234
1-greedy	0.315
LM baseline [29]	0.464
Cost-based, $\rho = 1.5$ [29]	0.429
Cost-based, $\rho = 5$ [29]	**0.465**

Table 7.7 lists the instance recall at rank ten results, along with instance recall scores computed on result sets from the subtopic retrieval work, corresponding to configurations presented in table 2 of Zhai et al.'s paper [29]. In their work, they looked at reranking a mixed pool of relevant and irrelevant documents drawn from the top documents selected by a language model baseline. For parity of comparison, we simulated their experiment conditions by reranking the same starting pool of documents as they did.

We note that 1-greedy outperforms our own PRP baseline, as we would expect. However, 1-greedy underperforms Zhai et al.'s system. Zhai et al.'s language model baseline appears to be a much better model for aspect retrieval than Naïve Bayes in the first place. If we had a well-tuned baseline, our 1-greedy would presumably perform better as well. Indeed, Zhai et al.'s reranking systems do not improve upon their baseline on instance recall, though this is probably due to their focus on optimizing the more sophisticated metrics of S-precision and WS-precision, and the (W)S-precision/S-recall curves.

7.5 Multiple Annotator Experiments

Another way of viewing the 1-call goal is from a multi-user perspective. Different users may intend different interpretations, as was evident from the Google examples presented earlier. For TREC-4 and TREC-6, multiple independent annotators were asked to make relevance judgments for the same set of topics, and over the same corpus [26, 14, 25]. In the TREC-4 case, these were topics 202 through 250, over a corpus of about 568,000 documents, and in the TREC-6 case, topics 301 through 350 over a corpus of about 556,000 documents (the TREC-6 topics are a subset of the robust track topics). TREC-4 had three annotators, TREC-6 had two.

Table 7.8: TREC-4, 6 results with multiple annotators.

TREC-4 (49 topics)				
Method	1-call (1)	1-call (2)	1-call (3)	1-call (total)
PRP	0.735	0.551	0.653	1.939
1-greedy	**0.776**	**0.633**	**0.714**	**2.122**
TREC-6 (50 topics)				
Method	1-call (1)	1-call (2)	1-call (3)	1-call (total)
PRP	0.660	0.620	N/A	1.280
1-greedy	**0.800**	**0.820**	N/A	**1.620**

The individual 1-call scores for each run and each annotator are presented in table 7.8. The last column is the sum of the previous columns, and can be considered to be the average number of annotators that are "satisfied" (that is, get at least one result they consider relevant in the top ten) by the respective result sets. Over both corpora, 1-greedy on average satisfied more annotators than PRP.

7.6 Query Analysis

To better understand 1-greedy's improvements, we also looked specifically at instances where 1-greedy returned a relevant result in the top ten (that is, satisfied the 1-call criterion) and PRP did not. The results for topic 100 and 113 from the TREC-1,2,3 weight-tuning development set is presented in figure 7-2 (document titles have been summarized for clarity and space), with the relevant result shaded.

Figure 7-2: TREC results where 1-greedy performs well.

Topic 100: Controlling the Transfer of High Technology

Rank	PRP	1-greedy
1	Data transfer software	Data transfer software
2	Disk controllers are getting smarter (SCSI and IDE)	Disk controllers are getting smarter (SCSI and IDE)
3	Caching hard-disk controllers (PC Week buyer's guide)	Environmental Protection Agency tech transfers
4	Environmental Protection Agency tech transfers	Wax vs. dye printers (PC Week buyer's guide)
5	Wax vs. dye printers (PC Week buyer's guide)	Engineering corporation tech transfer
6	Engineering corporation tech transfer	Serial-to-parallel network transfers
7	Department of Energy tech transfers	Whole-Earth technology (international tech transfer)
8	Serial-to-parallel network transfers	Department of Energy tech transfers
9	EISA and MCA technology	Fiber optic telecom line tech transfer through Soviet Union
10	Panel on tech transfer	Simon-Carves PCB tech transfer to Soviet Union

Topic 113: New Space Satellite Applications

Rank	PRP	1-greedy
1	10th anniversary of INTELSAT V	10th anniversary of INTELSAT V
2	US, Soviet Union spy satellites	US, Soviet Union spy satellites
3	FCC authorization of satellites	FCC authorization of satellites
4	Rescue of stranded satellite	NASA meeting agenda
5	Pentagon launch of spy satellite	Italian satellite failure
6	Satellite manufacture job market	Indian satellite problems
7	Indian satellite problems	Satellite cooperation in Europe
8	Italian satellite failure	New applications fuel satellite renaissance
9	Soviet satellite launch for paying customer	Space-based service station
10	Satellite for missile defense	Satellite on-board switching and processing

For topic 100, the TREC topic description is:

> Document will identify efforts by the non-communist, industrialized states to regulate the transfer of high-tech goods or "dual-use" technologies to undesirable nations.

It is notable that PRP wastes its time on the wrong interpretation of the title—looking for *technologies* that *control* data *transfer*, such as hard drive controllers. While 1-call also pulls up that interpretation, it moves away quickly enough that it can bring back more results on actual tech transfers, including the relevant Soviet Union-related result. [2]

For topic 113, the TREC topic description is:

> Document will report on non-traditional applications of space satellite technology.

As in the previous topic, the ability of 1-call to explore more interpretations leads to it finding a relevant result. PRP returns two documents on spy satellites, and two

[2]Result 10, on the PCB tech transfer to the Soviet Union, could possibly be judged relevant as well, but was not in the document judgments at all, indicating that it was never judged.

related to satellites of the Soviet Union, which effectively pushes down the relevant documents on novel satellite applications.

Chapter 8

Conclusions

In this thesis, we have studied the problem of probabilistic retrieval in a Bayesian context. We have shown shown how we can generalize the standard method of ranking by the probability of relevance, the probability ranking principle, to ranking by the expected value of a metric, the expected metric principle. We applied this ranking principle to a variety of metrics, both old and new, and derived ranking formulations for each. One specific metric that we have focused on in this thesis is k-call at rank n, which we believe captures both a notion of one-document relevance when $k = 1$, and perfect precision when $k = n$.

We have also shown that directly optimizing EMP objective functions can be intractable, in which case we can apply algorithmic heuristic search algorithms, such as the greedy algorithm, to reducing the search space.

Finally, we examined the empirical performance of these new greedy ranking algorithms, compared to a PRP baseline. We found that using the greedy approach applied to the 1-call and n-call metrics improved performance on 1-call and n-call, respectively. Incidentally, the natural greedy algorithm for optimizing for mean reciprocal rank and mean search length is the same as that for 1-call, so we also reported those scores and found improvement over the PRP baseline.

Much remains to be done to empirically explore heuristics that optimize our new, or other, objective functions. While the greedy approach performs reasonably well, we might expect more sophisticated techniques, such as local search algorithms, to

perform better.

Considering the particular metric of k-call, we have focused on the "extreme points" $k = 1$ and $k = n$. There is likely to be some value in filling in the middle. For example setting $k = 3$ says that a user wants several relevant documents but does not need them all to be relevant. As in the 1-call case, this would seem to allow the optimization algorithm to hedge—it has room, for example, to include 3 distinct interpretations in the top 10.

Our focus on an objective function means that our approach can theoretically be applied to *any* probabilistic model in which it is possible to discuss the likelihood of relevance of collections of documents. This includes, for example, the two-Poisson model [15], or the language modeling approach [20]. Those better models would hopefully yield better performance.

In general, our work indicates the potential value of "teaching to the test"—choosing, as the objective function to be optimized in the probabilistic model, the metric used to evaluate the information retrieval system. Assuming the metric is an accurate reflection of result quality for the given application, our approach argues that optimizing the metric will guide the system toward desired results.

Appendix A

Probability Background

This appendix briefly reviews some of the basic probabilistic machinery used in the rest of the thesis. For a more complete overview, see any standard probability text [2]. The discussion here is focused on discrete distributions and discrete random variables, but applies with some modifications for continuous distributions as well. The reader is assumed to be familiar with the definitions of probability and random variables.

A.1 Conditional Probabilities

The *conditional probability* of an event B given an event with nonzero probability A is defined as

$$\Pr[B \mid A] = \frac{\Pr[B \cap A]}{\Pr[A]}. \tag{A.1}$$

We call a set of events with nonzero probability $A_1, \ldots, A_n$ a *partition* of the event space if each pair of events is mutually exclusive (i.e., for all $i \neq j$, $\Pr[A_i \cap A_j] = 0$) and the events taken together cover the entire event space (i.e., $\Pr[\bigcup_i A_i] = 1$).

Given such a partition, the *law of total probability* states that

$$\Pr[B] = \sum_{i=1}^{n} \Pr[B \mid A_i] \Pr[A_i]. \tag{A.2}$$

Putting together the definition of conditional probability (equation A.1) and the

law of total probability (equation A.2), we can derive *Bayes' Rule*:

$$\Pr[A_j \mid B] = \frac{\Pr[B \mid A_j]\Pr[A_j]}{\sum_{i=1}^{n}\Pr[B \mid A_i]\Pr[A_i]}. \tag{A.3}$$

A.2 Expectations

The *expectation* (or *expected value*) of a discrete random variable X is defined as

$$E[X] = \sum_x xPr[X = x]. \tag{A.4}$$

In particular, note that for a *binary random variable* X that only takes on values $\{0, 1\}$, $E[X] = \Pr[X = 1]$.

Conditional expectations are defined analogously to conditional probabilities. The conditional expectation of a random variable X given an event with nonzero probability A is

$$E[X \mid A] = \sum_x xPr[X = x \mid A]. \tag{A.5}$$

Let $A_1, \ldots, A_n$ be a partition of the state space. Putting together the law of total probability (equation A.2) and the definition of conditional expectation (equation A.5), we arrive at the *law of total expectation*:

$$E[X] = E[E[X \mid A]] = \sum_{i=1}^{n} E[X \mid A_i]\Pr[A_i]. \tag{A.6}$$

Finally, for any set of random variables $X_1, \ldots, X_n$, the following *linearity of expectation* holds, regardless of dependencies between the variables:

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]. \tag{A.7}$$

Appendix B

Alternative Retrieval Models

In the body of this thesis we have focused strictly on probabilistic retrieval methods. Probabilistic methods are only one approach to the retrieval problem. In this appendix we present two other standard retrieval techniques, the Boolean and vector space models.

B.1 Boolean Models

Boolean models provide a simple, precise way of formalizing the retrieval problem. The query is expressed by the user as a Boolean statement in terms of the occurrence of index terms. Documents are simply represented by whether the index terms are present in them or not, without regards to occurrence frequency. For example, a query such as "driver AND (car OR NOT golf)" would retrieve documents with the term "driver", with the additional condition that the document should either have the term "car" or not have the term "golf". Intuitively, such a query may be used to find information about automobile drivers, while explicitly filtering out the golf club notion of driver.

The Boolean model has the obvious advantage of being a very simple, precise model. The disadvantages are, however, numerous. First, most users have difficulties expressing their information need in terms of the rather artificial construct of a Boolean logic statement. Usually, the query is assumed to be one long conjunction

or disjunction of the inputted search terms, which is a gross approximation of user information need. Second, there is no notion of ranking—either a document is relevant or irrelevant, with no intermediary notions of partial matching. Third, index term occurrence information is binary, throwing away a lot of possible information in the term frequency occurrence.

Due to these shortcomings, Boolean models may more appropriately be considered a data retrieval approach rather than an information retrieval approach. There is an implicit assumption that either a match is perfect (completely satisfying the information need) or completely irrelevant. This binary property is characteristic of finding data—and drives us toward exploring models that better capture the vast gray area between complete relevance and complete irrelevance.

B.2 Vector Space Models

Vector space models represent both documents and queries in a high dimensional space, and then match documents to the queries based on some distance measure. If we let V be the set of index terms (the *vocabulary*), then each document is a $|V|$-dimensional vector, where each element of the vector is some *weight* of that index term for that document. The query is itself also represented as such a vector. We then rank the documents by how close their vectors are to the query vector.

We have left two gaps in this model: the way index terms are weighted in the vectors, and the similarity metric. We examine each in turn.

Obviously, one way of setting the weights is to use a binary 0/1 assignment, reminiscent of what we did for Boolean retrieval. However, we can do better by using real-valued assignments. The most popular class of term weighting schemes is inspired by simple intuitions. First, we would expect that documents with many occurrences of an index term, should have a higher weight for that term. Second, some index terms are very common throughout the corpus, whereas others are very discriminative (that is, they appear only in a small subset of the corpus). The latter set of terms should be emphasized in the weighting, because they are what make the

document unique. The *text frequency-inverse document frequency* (tf-idf) methods are based on these intuitions. Let $\text{freq}(w, d)$ be the number of times term w occurs in document d, $\text{occurs}(w, d)$ be a binary function that is one if term w occurs in document d and zero otherwise, D be the set of all documents (the corpus), and V be the set of all index terms (the vocabulary). Then the tf of an index term w in a document d is the number of occurrences of that term, normalized by the count of the term that occurs most frequently in d:

$$\text{tf}(w, d) = \frac{\text{freq}(w, d)}{\max_{v \in V} \text{freq}(v, d)}. \tag{B.1}$$

Inverse document frequency (the idf factor) of a term w is the logarithm of the number of corpus documents over of the number of documents in which w appears:

$$\text{idf}(w) = \frac{|D|}{\sum_{d \in D} \text{occurs}(w, d)}. \tag{B.2}$$

tf is higher when a term occurs more frequently in a particular document, and idf favors terms that are more unique in the corpus overall—exactly the intuitions we wanted to capture. However, the question still remains of how we balance the tf and idf factors. The factors could simply be multiplied together to produce the final term weight:

$$\text{weight}(w, d) = \text{tf}(w, d) \times \text{idf}(w). \tag{B.3}$$

More sophisticated techniques may combine the factors in a more complex way. Finally, a document is just a vector of these weights, one weight per index term. The query vector can be calculated this way as well, or we may have a separate formula for it.

Now that we have vector representations of the documents and the query, the second problem is how we define a distance measure between them. The straightforward and commonly used distance measure is the *cosine of the angle* between the vectors. Recall that the cosine of the angle between two vectors is the vectors' dot product divided by the product of their Euclidean norms. Mathematically, if we let $\mathbf{d}$ be a

document vector and $\mathbf{q}$ the query vector, then the distance is

$$\text{distance}(\mathbf{d}, \mathbf{q}) = \frac{\mathbf{d} \cdot \mathbf{q}}{||\mathbf{d}|| \times ||\mathbf{q}||}. \tag{B.4}$$

(Note that $||\mathbf{x}||$ is the norm of vector $\mathbf{x}$.) We now can rank all the corpus documents against a query, according to descending order of query-document distance.

Vector space models are widely used in practice, due to their efficiency, simplicity, and good empirical relevance. However, the vector space models are rather *ad hoc*—the precise ranking formulas are difficult to theoretically justify. It is for this reason that we focus rely on a baseline probabilistic model for our work.

Appendix C

ML and MAP Estimators for the Multinomial Distribution

This appendix fills in the derivations for the maximum likelihood estimator for a multinomial distribution, and a maximum a posteriori estimator for a multinomial distribution with a Dirichlet prior.

C.1 Maximum Likelihood

Assume that we have m training examples, S_1 through S_m, that were sampled independently. Recall that the multinomial distribution is multivariate—that is, each sample of the distribution is a complete setting of n random variables, which we will call X_1 through X_n. Let x_i^j correspond to the ith dimension of the jth training example.

Recall the probability density function of the multinomial distribution (equation 3.6):

$$\mathrm{Mu}_n[X_1 = x_1, \ldots, X_n = x_n; \theta_1, \ldots, \theta_n] = \frac{(\sum_{i=1}^n x_i)!}{\prod_{i=1}^n x_i!} \prod_{i=1}^n \theta_i^{x_i}, \qquad \text{(C.1)}$$

and the general log likelihood function (equation 3.11):

$$\ell(\theta) = \sum_{i=1}^{m} \log \Pr[S_i; \theta]. \tag{C.2}$$

Plug the density equation into into the general log likelihood function to derive a specific log likelihood function for the multinomial distribution:

$$\begin{aligned}
\ell(\theta) &= \sum_{j=1}^{m} \log \mathrm{Mu}_n[S_j; \theta] \\
&= \sum_{j=1}^{m} \log \left(\frac{(\sum_{i=1}^{n} x_i^j)!}{\prod_{i=1}^{n} x_i^j!} \prod_{i=1}^{n} \theta_i^{x_i^j} \right) \\
&= \sum_{j=1}^{m} \log \left(\frac{(\sum_{i=1}^{n} x_i^j)!}{\prod_{i=1}^{n} x_i^j!} \right) + \sum_{j=1}^{m} \sum_{i=1}^{n} x_i^j \log \theta_i \\
&\stackrel{\text{mt}}{=} \sum_{i=1}^{n} \log \theta_i \sum_{j=1}^{m} x_i^j.
\end{aligned} \tag{C.3}$$

In the final step of this derivation, we dropped the first term of the likelihood function, since it is constant with respect to θ. We also have the additional constraints on θ that they must be positive and sum to one.

To maximize the log likelihood function, equation C.3, we need to use a Lagrange multiplier λ due to our constraint, which for convenience we will call $g(\theta)$, on the sum of θ:

$$g(\theta) = \sum_{i=1}^{n} \theta_i - 1 = 0. \tag{C.4}$$

Note that the positivity constraint can be accounted for as a boundary case. We take the partial derivative with respect to each parameter and set the results to be λ times

the partial derivative of the constraint. For the ith parameter, we find:

$$\begin{aligned}\frac{\partial \ell(\theta)}{\partial \theta_i} &= \lambda \times \frac{\partial g(\theta)}{\partial \theta_i} \\ \frac{\sum_{j=1}^{m} x_i^j}{\theta_i} &= \lambda \\ \theta_i &= \frac{\sum_{j=1}^{m} x_i^j}{\lambda}. \end{aligned} \tag{C.5}$$

Solving all n partial derivatives (equation C.5) and the constraint (equation C.4) together simultaneously, we find that the appropriate maximum likelihood estimate for parameter $1 \leq k \leq n$ is:

$$\hat{\theta}_k = \frac{\sum_{j=1}^{m} x_k^j}{\sum_{i=1}^{n} \sum_{j=1}^{m} x_i^j}. \tag{C.6}$$

C.2 Maximum A Posteriori

As before, assume that we have m training examples, S_1 through S_m, that were sampled independently.

Recall the posterior parameter likelihood equation (equation 3.17):

$$\Pr[\theta \mid S_1, \ldots, S_m] = \sum_{i=1}^{m} \log \Pr[S_i \mid \theta] + \log \Pr[\theta], \tag{C.7}$$

and the Dirichlet distribution probability function (equation 3.18):

$$\mathrm{Dir}_n[X_1 = x_1, \ldots, X_n = x_n; \alpha_1, \ldots, \alpha_n] = \begin{cases} \frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} x_i^{\alpha_i - 1} & \text{if } \sum_{i=1}^{n} x_i = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{C.8}$$

We can express the posterior probability in terms of the probability functions for

the multinomial and Dirichlet distributions (equations C.1 and C.8):

$$\begin{aligned}
&\Pr[S_1, \ldots, S_m \mid \theta] \Pr[\theta] \\
&= \sum_{j=1}^{m} \log \mathrm{Mu}_n[S_j; \theta] + \log \mathrm{Dir}_n[\theta; \alpha] && \text{(C.9)} \\
&\overset{\text{mt}}{=} \sum_{i=1}^{n} \log \theta_i \sum_{j=1}^{m} x_i^j + \log \left(\frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} \right) && \text{(C.10)} \\
&\overset{\text{mt}}{=} \sum_{i=1}^{n} \log \theta_i \sum_{j=1}^{m} x_i^j + \sum_{i=1}^{n} (\alpha_i - 1) \log \theta_i && \text{(C.11)} \\
&= \sum_{i=1}^{n} \log \theta_i \left(\sum_{j=1}^{m} x_i^j + \alpha_i - 1 \right) && \text{(C.12)}
\end{aligned}$$

Note we are considering the constraint $\sum_{i=1}^{n} \theta_i = 1$ separately, as we did in the maximum likelihood case. From equation C.9 to equation C.10, we used the derivation for the likelihood in equation C.3. We dropped constants in the next step (equation C.10 to C.11).

Note the similarity between our simplified expression for the posterior, equation C.12, and the simplified form for the likelihood function from maximum likelihood, equation C.3. Indeed, the only difference is the constant coefficient of the $\log \theta_i$ terms. Given that we are using the same θ constraint as well (equation C.4), we can find the maximum in exactly the same manner to obtain the following MAP estimate:

$$\hat{\theta}_k = \frac{\sum_{j=1}^{m} x_k^j + \alpha_i - 1}{\sum_{i=1}^{n} \sum_{j=1}^{m} x_i^j + \sum_{i=1}^{n} \alpha_i - n}. \qquad \text{(C.13)}$$

Appendix D

Sample Stopwords List

The following stopwords were filtered out from the corpora in our experiments.

```
a
and
are
as
at
be
but
by
for
if
in
into
is
it
no
not
of
on
```

```
or
s
such
t
that
the
their
then
there
these
they
this
to
was
will
with
1
2
3
4
5
6
7
8
9
0
```

Bibliography

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., 1999.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Introduction to Probability.* Athena Scientific, 2002.

[3] Abraham Bookstein. Information retrieval: A sequential learning process. *Journal of the American Society for Information Science (ASIS)*, 34(5):331–342, 1983.

[4] C. Buckley, G. Salton, and J. Allan. Automatic retrieval with locality information using smart. In *Proceedings of TREC-1*, pages 59–72, 1992.

[5] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of ACM SIGIR 1998*, pages 335–336, 1998.

[6] William S. Cooper. Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.

[7] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.

[8] Steve Cronen-Townsend and W. Bruce Croft. Quantifying query ambiguity. In *Proceedings of Conference on Human Language Technology (HLT 2002)*, pages 94–98, 2002.

[9] E.N. Efthimiadis. Query expansion. In *Annual Review of Information Systems and Technology*, pages 121–187, 1996.

[10] Jianfeng Gao, Haoliang Qi, Xinsong Xia, and Jian-Yun Nie. Linear discriminant model for information retrieval. In *Proceedings of ACM SIGIR 2005*, pages 290–297, 2005.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1983.

[12] Michael D. Gordon and Peter Lenk. A utility theoretic examination of the probability ranking principle in information retrieval. *Journal of the ASIS*, 42(10):703–714, 1991.

[13] Michael D. Gordon and Peter Lenk. When is the probability ranking principle suboptimal? *Journal of the ASIS*, 43(1):1–14, 1992.

[14] D. K. Harman. Overview of the fourth text retrieval conference (trec-4). In *Proceedings of TREC-4*, 1995.

[15] S. P. Harter. A probabilistic approach to automatic keyword indexing: Part i, on the distribution of specialty words in a technical literature. *Journal of the ASIS*, 26(4):197–206, 1975.

[16] William R. Hersh and Paul Over. Trec-8 interactive track report. In *Proceedings of TREC-8*, 1999.

[17] K. Spärck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments, part 1. *Information Processing and Management*, 36:779–808, 2000.

[18] John Lafferty and ChengXiang Zhai. Probabilistic relevance models based on document and query generation. *Language Modeling for Information Retrieval, Kluwer International Series on Information Retrieval*, 13, 2003.

[19] Anton V. Leouski and W. Bruce Croft. An evaluation of techniques for clustering search results. Technical Report IR-76, University of Massachusetts, Amherst, 1996.

[20] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of ACM SIGIR 1998*, pages 275–281, 1998.

[21] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML 2003*, pages 616–623, 2003.

[22] S. E. Robertson. The probability ranking principle in ir. In *Readings in information retrieval*, pages 281–286. Morgan Kaufmann Publishers Inc., 1997.

[23] Chirag Shah and W. Bruce Croft. Evaluating high accuracy retrieval techniques. In *Proceedings of ACM SIGIR 2004*, pages 2–9, 2004.

[24] Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of ACM SIGIR 2006*, pages 11–18, 2006.

[25] Ellen M. Voorhees. Overview of the sixth text retrieval conference (trec-6). In *Proceedings of TREC-6*, 1997.

[26] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of ACM SIGIR 1998*, pages 315–323, 1998.

[27] Ellen M. Voorhees. Measuring ineffectiveness. In *Proceedings of ACM SIGIR 2004*, pages 562–563, 2004.

[28] Ellen M. Voorhees. Overview of the trec 2004 robust retrieval track. In *Proceedings of TREC 2004*, 2004.

[29] ChengXiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of ACM SIGIR 2003*, pages 10–17, 2003.

[30] ChengXiang Zhai and John Lafferty. A risk minimization framework for information retrieval. In *Proceedings of the ACM SIGIR 2003 Workshop on Mathematical/Formal Methods in IR*, 2003.